onnet

2021
VIE

# Odoo ORM: Common ORM

www.on.net.vn
www.arrowhitech.com

onnet
BY AHT TECH JSC

# Table of contents



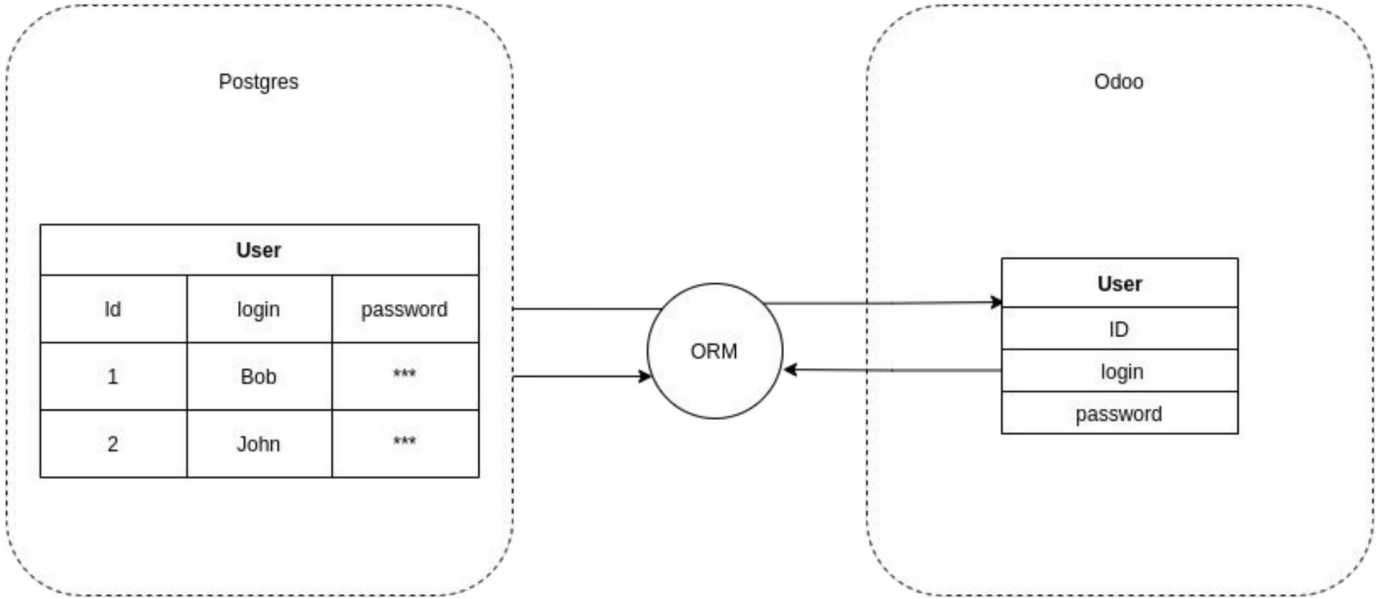

onnet
BY AHT TECH JSC

# Giới thiệu

Ánh xạ quan hệ-đối tượng (công cụ ánh xạ ORM, O / RM và O / R) trong khoa học máy tính là một kỹ thuật lập trình để chuyển đổi dữ liệu giữa các hệ thống kiểu không tương thích bằng ngôn ngữ lập trình hướng đối tượng.

Trên thực tế, điều này tạo ra một "cơ sở dữ liệu đối tượng ảo" có thể được sử dụng từ bên trong ngôn ngữ lập trình.

# Giới thiệu



# Bộ ghi

- Một bộ sưu tập có thứ tự các bản ghi của cùng một mô hình.

- Tương tác với các mô hình và bản ghi được thực hiện thông qua các tập bản ghi

# Bộ ghi



Các phương thức được định nghĩa trên một mô hình được thực thi trên một tập bản ghi và bản thân của chúng là một tập bản ghi:

```
class AModel (models.Model):

    _name = 'a.model'

    def a_method (tự):

        # self có thể là bất kỳ thứ gì giữa 0 bản ghi và tất cả các bản ghi trong cơ sở dữ liệu self.do_operation
        ()
```

# Bộ ghi

Lặp lại trên một tập bản ghi sẽ mang lại các tập hợp mới của một bản ghi (“các đĩa đơn”):

```
def do_operation (self): print
      (self) # => a.model (1, 2, 3, 4, 5)
      để ghi lại bản thân:
            print (record) # => a.model (1), sau đó là a.model (2), sau đó là a.model (3), ...
```

# Bộ ghi: Truy cập trường

Các tập bản ghi cung cấp giao diện "Bản ghi hoạt động" : các trường mô hình có thể được đọc và ghi trực tiếp từ bản ghi dưới dạng các thuộc tính.

Giá trị trường cũ ng có thể được truy cập như các mục dict, thanh lịch hơn và an toàn hơn getattr () cho tên trường động.

Việc đặt giá trị của trường sẽ kích hoạt cập nhật cơ sở dữ liệu:

```
>>> record.name
Tên ví dụ
>>> record.company_id.name
Tên công ty
>>> record.name = "Bob"
>>> field = "name"
>>> ghi [lĩnh vực]
Bob
```

# Bộ ghi: Ghi bộ nhớ cache và tìm nạp trước



Odoo duy trì một bộ đệm ẩn cho các trường của bản ghi, để không phải mọi truy cập trường đều tạo ra một cơ sở dữ liệu yêu cầu, điều này sẽ rất tệ đối với hiệu suất. Ví dụ sau chỉ truy vấn cơ sở dữ liệu cho tuyên bố đầu tiên:

```
record.name                             # quyền truy cập đầu tiên đọc giá trị từ cơ sở dữ liệu

record.name                             # lần truy cập thứ hai nhận giá trị từ bộ nhớ cache
```

# Bộ ghi: Ghi bộ nhớ cache và tìm nạp trước



Để tránh đọc một trường trên một bản ghi tại một thời điểm, Odoo tìm nạp trước các bản ghi và trường sau một số heuristics để có được hiệu suất tốt.

Hãy xem xét ví dụ sau, trong đó các đối tác là một tập bản ghi gồm 1000 bản ghi. Nếu không tìm nạp trước, vòng lặp sẽ thực hiện 2000 truy vấn đến cơ sở dữ liệu. Với tìm nạp trước, chỉ một truy vấn được thực hiện:

```
cho đối tác trong các đối tác:
    print partner.name                  # lần tìm nạp trước vượt qua đầu tiên 'tên' và 'lang'
    đối tác in .lang                    # (và các trường khác) trên tất cả các 'đối tác'
```

# Môi trường



Môi trường lưu trữ các dữ liệu ngữ cảnh khác nhau được ORM sử dụng: con trỏ cơ sở dữ liệu (đối với truy vấn cơ sở dữ liệu), người dùng hiện tại (để kiểm tra quyền truy cập) và ngữ cảnh hiện tại (lưu trữ siêu dữ liệu tùy ý ). Môi trường cũ ng lưu trữ bộ nhớ đệm.

Tất cả các tập bản ghi đều có một môi trường, là bất biến, có thể được truy cập bằng env và cấp quyền truy cập vào:

- người dùng hiện tại (người dùng) • con trỏ (cr) • cờ siêu người dùng (su) • hoặc ngữ cảnh (context)

Ví dụ:

```
>>> record.env

<Đối tượng môi trường ...>

>>> record.env.user

người dùng lại (3)

>>> record.env.cr

<Đối tượng con trỏ ...)
```

# Môi trường

Khi tạo một tập bản ghi từ một tập bản ghi khác, môi trường được kế thừa. Môi trường có thể được sử dụng để lấy một tập bản ghi trống trong một mô hình khác và truy vấn mô hình đó:

```
>>> self.env ['res.partner']

res.partner ()

>>> self.env ['res.partner']. 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)
```

# Môi trường: Thay đổi môi trường



`Model.with_context ([context] [, ** overrides])` bản ghi

Trả về phiên bản mới của tập bản ghi này được đính kèm với ngữ cảnh mở rộng.

Ngữ cảnh mở rộng là ngữ cảnh được cung cấp trong đó các ghi đè được hợp nhất hoặc ngữ cảnh hiện tại mà các ghi đè được hợp nhất, ví dụ:

```
# ngữ cảnh hiện tại là {'key1': True}
r2 = records.with_context ({}, key2 = True)
# -> r2._context là {'key2': True}
r2 = records.with_context (key2 = True)
# -> r2._context là {'key1': True, 'key2': True}
```

# Môi trường: Thay đổi môi trường

`Model.with_user (người dùng)`

Trả lại phiên bản mới của tập bản ghi này được đính kèm với người dùng nhất định, ở chế độ không phải siêu người dùng, trừ khi người dùng là siêu người dùng (theo quy ước, siêu người dùng luôn ở chế độ siêu người dùng).

`Model.with_company (công ty)`

Trả lại phiên bản mới của tập bản ghi này với ngữ cảnh đã sửa đổi, như sau:

```
result.env.company = công ty
result.env.companies = self.env.companies | Công ty
```

`Model.with_env (env)`

Trả lại phiên bản mới của tập bản ghi này được đính kèm với môi trường được cung cấp

`Model.sudo ([flag = True])`

Trả về phiên bản mới của tập bản ghi này với chế độ siêu người dùng được bật hoặc tắt, tùy thuộc vào cờ. Chế độ siêu người dùng không thay đổi người dùng hiện tại và chỉ bỏ qua kiểm tra quyền truy cập.

# Môi trường: Thực thi SQL



Thuộc tính cr trên môi trường là con trỏ cho giao dịch cơ sở dữ liệu hiện tại và cho phép thực thi SQL trực tiếp, đối với các truy vấn khó diễn đạt bằng ORM (ví dụ: các phép nối phức tạp) hoặc vì lý do hiệu suất:

```
self.env.cr.execute ("some_sql", params)
```

# Môi trường: Thực thi SQL



`Model.invalidate_cache (fnames = Không, id = Không)`

Vô hiệu hóa bộ nhớ cache của bản ghi sau khi một số bản ghi đã được sửa đổi. Nếu cả tên và id đều Không có, toàn bộ bộ nhớ cache sẽ bị xóa.

Thông số

- fnames - danh sách các trường đã sửa đổi hoặc Không có cho tất cả các trường
- id - danh sách id bản ghi đã sửa đổi hoặc Không có cho tất cả

# ORM chung: tạo


Model.create (vals_list) hồ sơ

Tạo các bản ghi mới cho mô hình.

Các bản ghi mới được khởi tạo bằng cách sử dụng các giá trị từ danh sách dicts vals_list và nếu cần, các giá trị đó từ default_get ().

Thông số

vals_list (danh sách) - giá trị cho các trường của mô hình, dưới dạng danh sách từ điển: [{'field_name': field_value, ...}, ...]

Để tương thích ngược, vals_list có thể là một từ điển. Nó được coi như một danh sách singleton [vals] và một bản ghi duy nhất được trả về.

Lợi nhuận

Các bản ghi đã tạo

# ORM chung: ghi



`Model.write (vals)`

Cập nhật tất cả các bản ghi trong tập hợp hiện tại với các giá trị được cung cấp.

Thông số

vals (dict) - các trường cần cập nhật và giá trị cần đặt trên chúng. ví dụ: `{'foo': 1, 'bar': "Qux"}` sẽ đặt trường foo thành 1 và thanh trường thành "Qux" nếu chúng hợp lệ (nếu không sẽ gây ra lỗi).

# ORM thông thường: sao chép


`Model.copy (mặc định = Không có)`

Bản ghi nhân bản tự cập nhật nó với các giá trị mặc định

Thông số

mặc định (dict) - từ điển các giá trị trường để ghi đè các giá trị ban đầu của bản ghi đã sao chép, ví dụ:
`{'field_name': overridden_value, ...}`

Lợi nhuận

kỷ lục mới

# ORM phổ biến: default_get

`Model.default_get (fields_list)` default_values

Trả về giá trị mặc định cho các trường trong danh sách_sảng. Giá trị mặc định được xác định bởi ngữ cảnh, mặc định của người dùng và chính mô hình.

Thông số

fields_list (danh sách) - tên của trường có yêu cầu mặc định

Lợi nhuận

từ điển ánh xạ tên trường với các giá trị mặc định tương ứng của chúng, nếu chúng có giá trị mặc định.

# ORM phổ biến: name_create

Model.name_create (tên) bản ghi

Tạo một bản ghi mới bằng cách gọi create () chỉ với một giá trị được cung cấp: tên hiển thị của bản ghi mới.

Bản ghi mới sẽ được khởi tạo với bất kỳ giá trị mặc định nào áp dụng cho mô hình này hoặc được cung cấp thông qua ngữ cảnh. Hành vi thông thường của create () áp dụng.

Thông số

name - tên hiển thị của bản ghi để tạo

Lợi nhuận

the name get () giá trị cặp của bản ghi đã tạo

# ORM chung: duyệt

Model.browse ([id])  bản ghi

Trả về tập bản ghi cho các id được cung cấp dưới dạng tham số trong môi trường hiện tại.

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

Thông số

id (int hoặc danh sách (int) hoặc Không) - (các) id

Lợi nhuận

bộ hồ sơ

# ORM phổ biến: tìm kiếm



Model.search (args [, offset = 0] [, limit = None] [, order = None] [, count = False]) [source]

Tìm kiếm các bản ghi dựa trên miền tìm kiếm args.

Thông số

- args - Miền tìm kiếm. Sử dụng danh sách trống để khớp với tất cả các bản ghi.
- offset (int) - số lượng kết quả cần bỏ qua (mặc định: không có)
- giới hạn (int) - số lượng bản ghi tối đa để trả về (mặc định: tất cả)
- order (str) - sắp xếp chuỗi
- số đếm (bool) - nếu Đúng, chỉ đếm và trả về số lượng bản ghi phù hợp (mặc định: Sai)

Lợi nhuận

giới hạn tối đa các bản ghi phù hợp với tiêu chí tìm kiếm

# ORM phổ biến: search_count

Model.search_count (args) __

int Trả về số lượng bản ghi trong mô hình hiện tại khớp với miền được cung cấp.

# ORM phổ biến: name_search



Model.name_search (name = '', args = None, operator = 'ilike', limit = 100) bản ghi

Tìm kiếm các bản ghi có tên hiển thị khớp với mẫu tên đã cho khi so sánh với toán tử đã cho, đồng thời khớp với miền tìm kiếm tùy chọn (args).

Ví dụ, điều này được sử dụng để cung cấp các đề xuất dựa trên giá trị một phần cho trường quan hệ.
Đôi khi được xem như là hàm ngược của name_get (), nhưng nó không được đảm bảo là.

Phương thức này tương đương với việc gọi search () với một tên miền tìm kiếm dựa trên tên_người_ hiển thị và sau đó tên mục chính chính () trên kết quả của việc tìm kiếm.

Thông số

- tên (str) - mẫu tên phù hợp • args (danh sách) - miền tìm kiếm tùy chọn (xem tìm kiếm () cho cú pháp), chỉ định các hạn chế khác • toán tử (str) - toán tử miền cho tên phù hợp, chẳng hạn như 'like' hoặc '='. • giới hạn (int) - số lượng bản ghi tối đa tùy chọn để trả về

Lợi nhuận

danh sách các cặp (id, text_repr) cho tất cả các bản ghi phù hợp.

# ORM chung: đọc

Model.read ([lĩnh vực])

Đọc các trường được yêu cầu cho các bản ghi theo phương pháp tự, cấp thấp / RPC. Trong mã Python, ưu tiên duyệt ().

Thông số

các trường - danh sách các tên trường cần trả về (mặc định là tất cả các trường)

Lợi nhuận

danh sách các từ điển ánh xạ tên trường với giá trị của chúng, với một từ điển cho mỗi bản ghi

# ORM chung: đã lọc



`Model.filtered` (func)

Trả lại hồ sơ trong func tự thỏa mãn.

Thông số

func (có thể gọi hoặc str) - một hàm hoặc một chuỗi tên trường được phân tách bằng dấu chấm

Lợi nhuận

tập các bản ghi thỏa mãn func, có thể trống.

Ví dụ: #

```
chỉ giữ các bản ghi có công ty là bản ghi của người dùng hiện tại.filtered
(lambda r: r.company_id == user.company_id)

# chỉ lưu giữ hồ sơ có đối tác là công ty
Records.filtered ("partner_id.is_company")
```

# ORM phổ biến: được ánh xạ



`Model.mapped (func)`

Tự áp dụng func trên tất cả các bản ghi và trả về kết quả dưới dạng danh sách hoặc tập bản ghi (nếu func trả về bộ hồ sơ). Trong trường hợp thứ hai, thứ tự của tập bản ghi được trả về là tùy ý.

Thông số

func (có thể gọi hoặc str) - một hàm hoặc một chuỗi tên trường được phân tách bằng dấu chấm

Lợi nhuận

Trả về danh sách tổng hai trường cho mỗi bản ghi trong tập hợp:

```
Records.mapped (lambda r: r.field1 + r.field2)
```

Hàm được cung cấp có thể là một chuỗi để nhận các giá trị trường:

```
# trả về danh sách tên
Records.mapped ('name') #
trả về tập hợp các bản ghi của các đối
tác record.mapped ('partner_id') # trả
về liên hợp của tất cả các ngân hàng đối tác, với các bản sao đã bị loại bỏ
records.mapped ('partner_id.bank_ids')
```

# Q&A