

## 1 HARI 2: INHERITANCE – INSTRUCTOR

Halo brooooo.... masih semangat???

Hari ini kita mulai masih ke materi inheritance di odoo. Contoh kasus kita turunkan class `res.partner` yang tersedia di odoo dan dimanfaatkan sebagai Instruktur pada addons Academic.

Disini kita tambahi fitur class `res.partner` dengan satu field tambahan `is_instructor` bertype `Boolean`. Nantinya Partner yang bertype instruktur bisa dipilih di form Session, dan jika tidak maka tidak bisa dipilih di Session sebagai Instruktur.

### 1.1 INHERIT PARTNER CLASS

Untuk menurunkan (inherit) suatu class, bikin file baru dengan nama file `partner.py`.

Isinya seperti ini...

```
from odoo import api, fields, models, _  
  
class Partner(models.Model):  
    _name = 'res.partner'  
    _inherit = 'res.partner'  
  
    is_instructor = fields.Boolean(string="Is Instructor", )
```

*Gambar 1 Tambahi field is\_instructor di res.partner*

Disini kita lihat cara meng-inherit class `res.partner`, yaitu dengan membuat atribut `_name` sama seperti induknya supaya penambahan fitur yang kita lakukan bergabung pada class induknya.

Lalu pakai juga atribut `_inherit` yang menandakan dari mana class ini diturunkan yaitu class `res.partner`.

Udah itu kita tambahkan field `is_instructor` pada atribut `_columns` seperti biasa.

Setelah proses ini maka class `res.partner` asli akan bertambah fieldnya.

Kalo udah beres panggil lewat `__init__.py` seperti biasa.

```
import session
import course
import attendee
import partner
```

*Gambar 2 Import partner.py dari \_\_init\_\_.py*

Sejauh ini struktur file addons kita adalah sebagai berikut... ada tambahan file `partner.py`.

```
academic
|-- __init__.py
|-- __openerp__.py
|-- attendee.py
|-- attendee.xml
|-- course.py
|-- course.xml
|-- menu.xml
|-- partner.py
|-- session.py
`-- session.xml
```

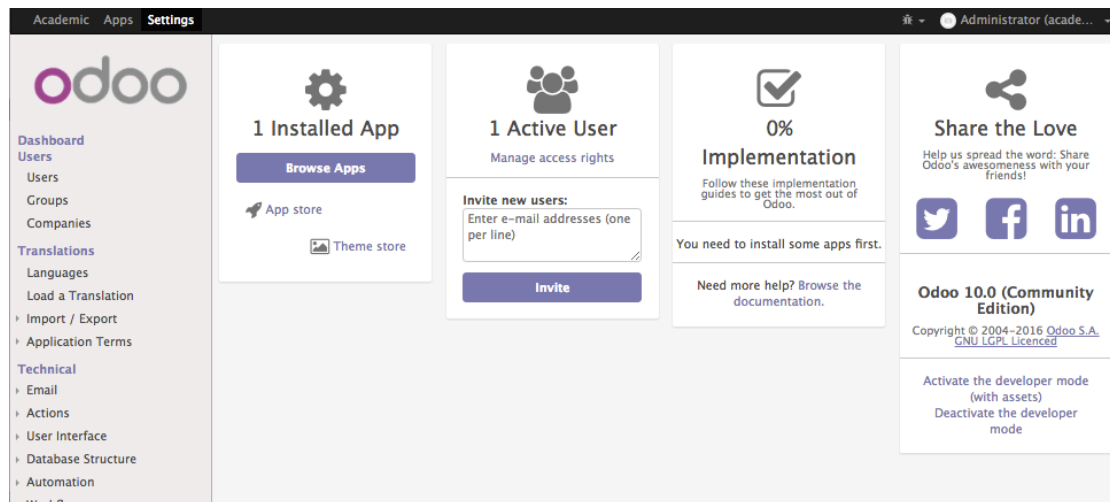
Setelah proses inherit ini, maka field `is_instructor` udah tersedia pada class asli `res.partner`.

Untuk munculinnya kita perlu modif view partner bawaan odoo yaitu dengan meng-inherit view.

## 1.2 INHERIT PARTNER VIEW

Kita perlu inherit view form Partner bawaan odoo untuk bisa menampilkan field tambahan `is_instructor` tadi.

Pertama-tama kita perlu tau, apa nama form partner yang sekarang ada, caranya aktifkan Developer Mode (dari menu `Settings > Activate Developer Mode`)...

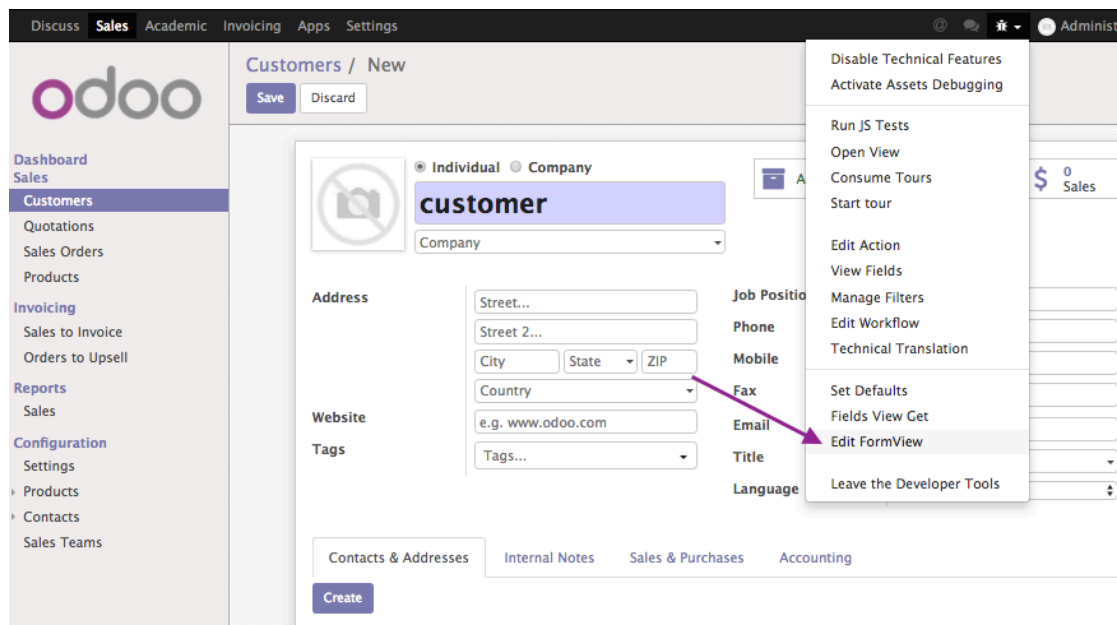


Gambar 3 Activate Developer Mode

Klik Activate the Developer Mode.

Buka form Partner dari menu **Sale - Customer**. Install module Sale Management jika belum ada menu tersebut.

Di bagian kiri atas ada pilihan **Debug View**.. pilih **Edit Form View**.



Gambar 4 Lihat definisi form view

Muncul definisi form dari form yang sedang dibuka yaitu Partner form.

**Edit FormView**

View Name:

View Type:

Model:

Sequence:

Active: ☒

Child Field:

Inherited View:

View inheritance mode:

Model Data:

External ID:

Architecture | Access Rights | Inherited Views

Edit Translations

```
1 <?xml version="1.0"?>
2 <form string="Partners">
3   <sheet>
4     <div class="oe_button_box" name="button_box">
5       <button name="toggle_active" type="object" class="oe_stat_button" icon="fa-archive">
6         <field name="active" widget="boolean_button" options="{&quot;terminology&quot;: &quot;archive&quot;}" />
7       </button>
8     </div>
9     <div class="oe_avatar">
10      <field name="image" widget="image" class="oe_avatar" options="{&quot;preview_image&quot;: &quot;image_m&quot;}" />
11    </div>
12    <div class="oe_title">
13      <field name="is_company" invisible="1" />
14      <field name="commercial_partner_id" invisible="1" />
15      <field name="company_type" widget="radio" class="oe_edit_only" options="{&quot;horizontal&quot;: true}" />
16    </div>
17    <div class="oe_edit_only">
18      <field name="name" default_focus="1" placeholder="Name" attrs="{&quot;required&quot;: [(&quot;type&quot;, &quot;=&quot;, &quot;co&quot;)]}" />
19    </div>
20    <div class="o_row">
21      <field name="parent_id" placeholder="Company" domain="[(&quot;is_company&quot;, &quot;=&quot;, True)]" context="{&quot;default&quot;: &quot;parent_id&quot;}" />
22      <field name="company_name" attrs="{&quot;invisible&quot;: [(&quot;parent_id&quot;, &quot;=&quot;, False), (&quot;company_name&quot;, &quot;=&quot;, False)]}" />
23      <button name="create_company" type="object" string="Create company" class="btn btn-sm oe_edit_only" />
24    </div>
25  </sheet>
26 </form>
```

Save Discard

Gambar 5 Definisi form view res.partner

Dari sini kita tau bahwa id dari partner form adalah `base.view_partner_form`, dari field External ID form itu.

Setelah dapat id nya bikin turunan dari form view tadi dengan cara berikut.

Bikin file XML baru namanya `partner.xml` dibawah folder `academic`.

Isinya seperti berikut.

```

1  <openerp>
2      <data>
3          <record id="partner_instructor" model="ir.ui.view">
4              <field name="name">partner.instructor</field>
5              <field name="model">res.partner</field>
6              <field name="inherit_id" ref="base.view_partner_form" />
7              <field name="arch" type="xml">
8                  <field name="website" position="after">
9                      <field name="is_instructor" />
10                 </field>
11             </field>
12         </record>
13     </data>
14 </openerp>

```

Gambar 6 Nampilin `is_instructor` di `partner view form`

Disini intinya kita membuat record pada model `ir.ui.view` sama seperti waktu mau membuat form view biasa.

Atribut `id` dan Field `name` diisi dengan `res.partner.instructor.form` supaya membedakan dengan form view aslinya.

Field `model` diisi dengan sumber data form ini, yaitu `res.partner` yang merupakan object bawaan odoo.

Field `inherit_id`, ini yang paling penting. Menentukan dari view manakan view yang sekarang ini diturunkan, yaitu `base.view_partner_form`.

Field `arch` berisi definisi modif yang kita lakukan terhadap view asli, yaitu mengubah field `website` (field bawaan di object `res.partner`) dan menambahkan field baru pada posisi setelahnya (atribut `position="after"`).

Field yang kita tambahkan adalah field `is_instructor` yang merupakan field yang kita tambahkan pada object `res.partner` asli sebelumnya.

Tambahkan file XML ini kedalam `__openerp__.py`.

```
{
  "name": "Academic Information System Day 2",
  "version": "1.0",
  "depends": [
    "base",
  ],
  "author": "akhmad.daniel@gmail.com",
  "category": "Education",
  'website': 'http://www.vitraining.com',
  "description": """\
Academic Information System Day2
""",
  "data": [
    "menu.xml",
    "course.xml",
    "session.xml",
    "attendee.xml",
    "partner.xml",
  ],
  "installable": True,
  "auto_install": False,
  "application": True,
}
```

*Gambar 7 Panggil partner.xml dari \_\_openerp\_\_.py*

Pada step ini struktur file addon `academic` harus seperti ini...  
ada tambahan file `partner.xml`.

```
academic
|-- __init__.py
|-- __openerp__.py
|-- attendee.py
|-- attendee.xml
|-- course.py
|-- course.xml
|-- menu.xml
|-- partner.py
|-- partner.xml
|-- session.py
`-- session.xml
```

Restart odoo dan update module.

Hasilnya:

Gambar 8 Muncul Is Instructor di form Partner

Field `is_instructor` sudah muncul pada posisi yang diinginkan yaitu setelah field `website`.

### 1.3 MODIF FORM SESSION – FILTER PARTNER IS INSTRUCTOR

Lanjut... kita perlu manfaatin field `is_instructor` yang sekarang udah ada pada object `res.partner` untuk mem-filter Partner nama yang boleh muncul dan mana yang nggak boleh pada form Session.

Partner yang punya `is_instructor = False` nggak boleh muncul di pilihan instruktur pada form Session, demikian sebaliknya.

Modif file `session.xml`, edit bagian form view Session...

```

<record id="view_academic_session_form" model="ir.ui.view">
  <field name="name">academic.session.form</field>
  <field name="model">academic.session</field>
  <field name="type">form</field>
  <field name="priority" eval="8"/>
  <field name="arch" type="xml">
    <form string="Session">
      <sheet>
        <div class="oe_title">
          <label for="name" class="oe_edit_only"
            string="Session Name"/>
          <h1><field name="name"/></h1>
        </div>
        <group>
          <group>
            <field name="course_id" />
            <field name="instructor_id"
              domain="[('is_instructor','=',True)]"/>
            <field name="start_date" />
          </group>
          <group>
            <field name="duration" />
            <field name="seats" />
          </group>
        </group>
      </sheet>
    </form>
  </field>
</record>

```

Gambar 9 Tambahi domain di insturctor\_id

Disini kita nambahin atribut domain pada field **is\_instructor**.

Atribut domain disini isinya adalah notasi untuk mem-filter record yang boleh muncul pada field **instructor\_id**, ditulis dalam notasi Polish.

Bentuknya [ ( 'nama\_field','operasi','nilai') ]

Misalnya [ ('is\_instructor' , '=' , True) ]

Artinya pada pilihan drop down list field **instructor\_id** yang muncul hanya record yang sesuai kriteria domain, yaitu jika partner **is\_instructor** adalah True.

Akibatnya, hanya partner yang **is\_instructor = True** saja yang bisa dipilih di Session form.

Restart odoo dan update module.

Hasilnya...



Daftar Session / New

Save Discard

Session Name  
**Session1**

Course

Instructor

Start Date

Duration

Seats

Active ☐

Attendees

Name	Partner
Add an item	

Gambar 10 Hanya partner yang `is_instructor` True yang bisa dipilih

#### 1.4 MODIF FORM SESSION – FILTER PARTNER CATEGORY


Sekarang coba kita tambahi kriteria filter nya.. bukan hanya berdasarkan field `is_instructor` tapi kita juga mau pake `Category` dari Partner.

Misalnya kita masukkan beberapa Partner sebagai Category Partner "Pelatih". Kita mau selain dari `is_instructor`, semua Partner yang termasuk Category "Pelatih" juga boleh muncul di form Session.

Edit data beberapa partner, tambahin sebagai Category "Pelatih".

Customers / New

Save
Discard



Individual
Company

Joko

Company

Active

0.0  
Inv

Address

Street...

Street 2...

City
State
ZIP

Country

Website

e.g. www.odoo.com

Is Instructor

☐

Tags

Pelatih x Tags...

Job Position

e.g. Sal

Phone

Mobile

Fax

Email

Title

Language

English

Contacts & Addresses
Internal Notes
Sales & Purchases
Accounting

Create

Gambar 11 Tambahin kategori partner sebagai Pelatih tapi bukan Instructor

Customers / New

Save Discard

☒ Individual
 ☐ Company

Ujang

Company

Active 0.00 Invoiced

Address

Street...

Street 2...

City State ZIP

Country

Website

e.g. www.odoo.com

Is Instructor

Tags

Pelatih x Supervisor x

Tags...

No results to show...

Job Position

e.g. Sales Director

Phone

Mobile

Fax

Email

Title

Language

English

Contacts & Addresses Internal Notes Sales & Purchases Accounting

Create

Gambar 12 Cara set kategori partner

Lalu buka dan edit file `session.xml`, kita tambahi nilai attribute `domain` di field `instructor_id` form view Session. Seperti ini...

```
<group>
  <field name="course_id" />
  <field name="instructor_id"
        domain="[('id','is_instructor','=',True),
                  ('category_id','=', 'Pelatih')]" />
  <field name="start_date" />
</group>
```

Gambar 13 Tambahi domain kategori

Disini kita tambahi operator or " | " pada domain, yaitu jika `is_instructor=True` ATAU `category_id="Pelatih"`, maka record itu boleh muncul pada pilihan Instructor di form view Session.

Syntax pemulisanannya sama seperti sebelumnya, hanya saja operatornya ditaro di depan.

Bentuknya `[ '|', (kondisi1) , (kondisi2)]`

Misalnya `[ '|', ('is_instructor' , '=' , True), ('category_id' , '=' , 'Pelatih')]`

Akibatnya, partner yang bisa dipilih di Session form adalah yang punya `is_instructor = True` atau kategori "Pelatih".

Restart odoo dan update module.

Hasilnya...

Daftar Session / New

Save Discard

Session Name  
**session1**

Course

Instructor

Start Date

Duration

Seats

Active ☐

Attendees

Name	Partner
Add an item	

Gambar 14 Partner yang kategori Pelatih bisa dipilih sebagai instruktur

## 1.5 MENU INSTRUKTUR

Sekarang kita bikin menu dibawah menu Academic untuk menampilkan daftar Partner yang diset sebagai instruktur. Judul menu nya Instruktur, action window ke daftar Partner tapi udah otomatis terfilter sesuai is\_instructor = True atau Category "Pelatih".

Pertama, edit file `menu.xml` untuk nambahin menu Instruktur. Menu untuk waktu di klik akan membuka action window `action_instructor_list`.

```
<menuitem id="menu_academic_attendee"
  name="Attendee"
  parent="academic_1"
  action="action_attendee_list"
  sequence="40"/>

<menuitem id="menu_academic_instructor"
  name="Instructor"
  sequence="50"
  action="action_instructor_list"
  parent="academic_1"/>
```

```
</data>
</odoo>
```

Gambar 15 Tambah menu Instructor

Terus, bikin deklarasi action window di atas deklarasi menu Instructor, masih di file `menu.xml`, seperti ini...

```
<record id="action_instructor_list" model="ir.actions.act_window">
  <field name="name">Instructors</field>
  <field name="type">ir.actions.act_window</field>
  <field name="res_model">res.partner</field>
  <field name="view_type">form</field>
  <field name="view_mode">kanban,tree,form</field>
  <field name="context">{"search_default_instructor":1}</field>
  <field name="search_view_id" ref="base.view_res_partner_filter"/>
  <field name="help" type="html">
    <p class="oe_view_nocontent_create">
      Click to add an Instructor
    </p><p>
      tambah instructor
    </p>
  </field>
</record>
```

Gambar 16 Action window Partner Instructor

Kita lihat di atas, ada field context yang diisi dengan `"search_default_instructor":1`. ini maksudnya waktu action window dibuka pertama kali, otomatis dia akan mem-filter secara default dengan filter yang name nya adalah `instructor` pada search view `res.partner`. Ini perlu kita tambahi abis ini, karena filter `instructor` itu belum ada sekarang.

Lalu edit file `partner.xml` dalam rangka untuk meng-inherit search view `res.partner`.

```
<record id="view_res_partner_filter2" model="ir.ui.view">
  <field name="name">res.partner.select2</field>
  <field name="model">res.partner</field>
```

```

<field name="inherit_id" ref="base.view_res_partner_filter"/>
<field name="arch" type="xml">
  <search string="Search Partner">
    <filter string="Instructors"
      name="instructor" domain="[('is_instructor','=',1)]"
      help="Instructor Partners"/>
  </search>
</field>
</record>

</data>
</openerp>

```

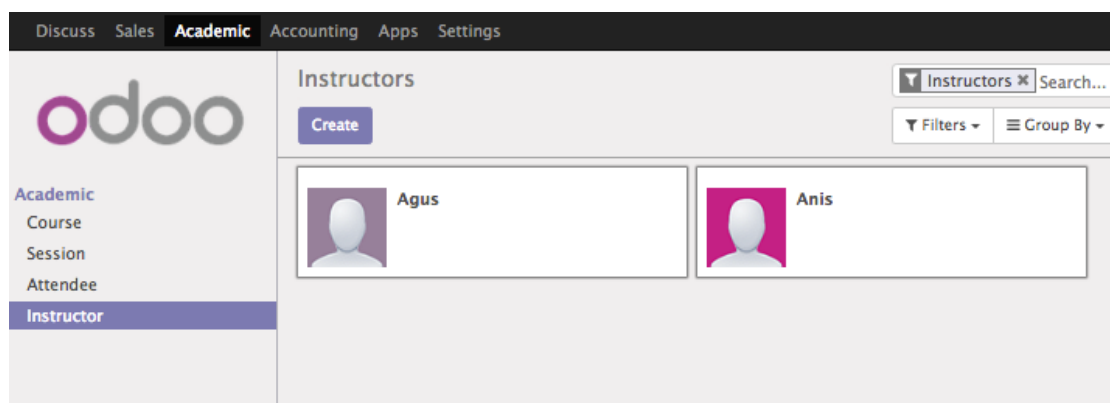
Gambar 17 Default filter partner instructor

Disini kita inherit view dengan id `base.view_res_partner_filter` yang merupakan view bawaan odoo untuk mem-filter data Partner.

Lalu kita tambahi filter baru yaitu dengan `name="instructor"` dan label `Instructors` dengan domain `is_instructor = 1`. Penambahan filter ini memungkinkan filter aktif secara default ketika dipanggil dari action window dengan context variable `"search_default_instructor":1`.

Restart odoo dan update module.

Hasilnya...



Gambar 18 Menu Instructor muncul dan membuka Partner dengan filter Instructors





## 2 FUNCTIONAL FIELDS – PERCENTAGE TAKEN SEATS

Gimana kalo mau tau berapa persen okupansi suatu session yang dihitung berdasarkan berapa jumlah peserta dibandingkan dengan jumlah seats ?

Angka itu harus dihitung secara real time jadi nggak disimpan di table database. Ini namanya functional field, mirip kayak calculated field atau virtual field di bahasa lain.

### 2.1 DEFINISI FUNCTION FIELDS

Buka file `session.py`, tambahkan satu field baru yang namanya `taken_seats`. Pendefinisianannya sama persis seperti field lain di Odoo tapi disini kita tambahi parameter `compute`, yaitu nama function untuk menghitung nilainya...

```
class session(models.Model):
    _name = 'academic.session'

    name = fields.Char("Name", required=True)

    course_id = fields.Many2one(comodel_name="academic.course",
                                string="Course", required=False, )
    instructor_id = fields.Many2one(comodel_name="res.partner",
                                    string="Instructor", required=False, )
    start_date = fields.Date(string="Start Date", required=False, )
    duration = fields.Integer(string="Duration", required=False, )
    seats = fields.Integer(string="Seats", required=False, )
    active = fields.Boolean(string="Active", )

    attendee_ids = fields.One2many(comodel_name="academic.attendee",
                                   inverse_name="session_id",
                                   string="Attendees",
                                   required=False, )

    taken_seats = fields.Float(compute="_calc_taken_seats",
                              string="Taken Seat", required=False, )
```

Gambar 19 Tambah function field `taken_seats`

Selanjutnya kita perlu bikin definisi function yang dipanggil oleh `taken_seats` parameter `compute`.

Dalam hal ini nama function-nya harus sama dengan yang dipanggil dari `compute`, yaitu `_calc_taken_seats`. Format penamaan pake underscore ini cuman konvensi aja untuk nandain bahwa ini private function.

Terus, kalau functional field, otomatis odoo akan kirim parameter function standard seperti ini:

```
nama_function(self)
```

- self = record-record dari current object yang hendak dihitung field taken\_seats nya , yaitu record-record Session

Berikut ini definisi function field-nya.

```
@api.depends('attendee_ids','seats')
def _calc_taken_seats(self):
    for rec in self:
        if rec.seats>0:
            rec.taken_seats = 100.0 * len(rec.attendee_ids)/rec.seats
        else:
            rec.taken_seats = 0.0
```

*Gambar 20 Definisi function \_calc\_taken\_seats()*

Record-record yang nilai function fieldnya mau dicari dimasukkan melalui parameter array list `self`. Walaupun yang dicari berupa satu record, tapi odoo tetap ngasi record-nya dalam bentuk array list, jadi nanti kita harus proses secara array list juga.

Kita tinggal looping variable self untuk membaca dan menulis nilai field setiap record:

```
for rec in self:
```

dalam setiap loop kita simpan ke dalam variable `rec` yang udah berupa satu record session, jadi tinggal diambil field-fieldnya dengan notasi dot.

```
for rec in self:
    if rec.seats>0:
        rec.taken_seats = 100.0 * len(rec.attendee_ids)/rec.seats
    else:
        rec.taken_seats = 0.0
```

Dalam hal ini kita cek dulu apakah `rec.seats` ada nilainya.

Kalo nggak ada langsung diset `rec.taken_seats = 0.0`.

Tapi kalau ada, kita set nilai `rec.taken_seats` dengan prosentase dari banyaknya peserta yang hadir (`len(session.attendee_ids)`) dibagi dengan jumlah seats (`session.seats`).

Hasil pembagian masih berupa integer karena masing-masing berupa integer. Untuk menjadikannya float sesuai type field functional, harus dikalikan dengan float, dalam hal ini kita kali dengan 100.0.

Dalam setiap perhitungan di dalam looping, nilainya langsung ditampung dalam variable `rec.taken_seat`.

Agar kita bisa membaca isi dari field lain pada perhitungan, di atas nama function kita tambahkan API decoration depends:

```
@api.depends('attendee_ids','seats')
def _calc_taken_seats(self):
```

Contohnay disini kita perlu membaca nilai dari `attendee_ids` dan `seats`.

## 2.2 MENAMPILKAN DI TREE DAN VIEW

Field functional dapat diperlakukan sama seperti field regular lainnya. Bisa ditampilkan di tree dan form view.

Edit `session.xml`, tambahi field `taken_seats` di form view setelah field `active`.

```
<group>
  <field name="duration" />
  <field name="seats" />
  <field name="active" />
  <field name="taken_seats" />
</group>
```

```

</group>
<notebook>
  <page string="Attendees">
    <field name="attendee_ids">
      <tree string="Attendees">
        <field name="name" />
        <field name="partner_id" />
      </tree>
    </field>
  </page>
</notebook>

```

Gambar 21 Tambah field taken\_seats di form view Session

Tambahi juga di tree view setelah field `active`.

```

<openerp>
  <data>
    <record id="view_academic_session_tree" model="ir.ui.view">
      <field name="name">academic.session.tree</field>
      <field name="model">academic.session</field>
      <field name="type">tree</field>
      <field name="priority" eval="8"/>
      <field name="arch" type="xml">
        <tree string="Session">
          <field name="course_id" />
          <field name="name"/>
          <field name="instructor_id" />
          <field name="start_date" />
          <field name="duration" />
          <field name="seats" />
          <field name="active" />
          <field name="taken_seats" />
        </tree>
      </field>
    </record>

```

Gambar 22 Tambah field taken\_seats di tree/list view Session

Restart odoo dan update module, hasilnya...

Field `taken_seats` muncul pada tree view Session.

Daftar Session

Search...

Create

Import

Filters

Group By

Favorites

1-1 / 1

<input type="checkbox"/>	Course	Name	Instructor	Start Date	Duration	Seats	Active	Taken Seat
<input type="checkbox"/>		Session1			0	100	<input checked="" type="checkbox"/>	2.00

Gambar 23 Muncul field taken\_seats di list view

dan muncul juga pada form view Session.

Daftar Session / Session1		<input type="button" value="Edit"/> <input type="button" value="Create"/>		<input type="button" value="Action"/>		1 / 1 <input type="button" value="Previous"/>	
---------------------------	--	---	--	---------------------------------------	--	---	--

### Session1

Course		Duration	0
Instructor		Seats	100
Start Date		Active	<input checked="" type="checkbox"/>
		Taken Seat	2.00

Name	Partner
01	Agus
02	Badu

Gambar 24 Muncul field taken\_seats di form view

Nilainya sudah sesuai dengan perhitungan total jumlah peserta / total seats \* 100%.

## 2.3 PROGRESSBAR DI FORM

Field taken\_seats boleh dimunculkan dengan tampilan khusus yang disebut progress bar. Supaya tampilannya bukan berupa angka tapi prosentase progress dari 100%.

Edit `session.xml`, tambahi attribute `widget="progressbar"` pada field tambahan `taken_seats` di form view.

```
<group>
  <group>
    <field name="course_id" />
    <field name="instructor_id"
domain="['|',('is_instructor','=',True),('category_id','=', 'Pelatih')]" />
    <field name="start_date" />
  </group>
  <group>
    <field name="duration" />
    <field name="seats" />
    <field name="active" />
    <field name="taken_seats" widget="progressbar"/>
  </group>
</group>
```

Gambar 25 Nampilin taken\_seats di form dengan progress bar

Restart odoo dan update module, hasilnya...

The screenshot shows the Odoo 'Daftar Session / Session1' form. At the top, there are 'Save' and 'Discard' buttons and a page indicator '1 / 1'. The form fields include 'Session Name' (filled with 'Session1'), 'Course', 'Instructor', and 'Start Date' (all dropdown menus), 'Duration' (input field with '0'), 'Seats' (input field with '10'), 'Active' (checkbox checked), and 'Taken Seat' (displayed as a blue progress bar at 20%). Below these fields is a table titled 'Attendees' with columns 'Name' and 'Partner'. The table contains two rows: '01' with partner 'Agus' and '02' with partner 'Badu'. Each row has a trash icon. At the bottom of the table is an 'Add an item' button.

Name	Partner
01	Agus
02	Badu

Gambar 26 Muncul progressbar persentasi seats di form

Field `taken_seats` sudah muncul di form view sebagai progress bar.

## 2.4 PROGRESS BAR DI TREE VIEW

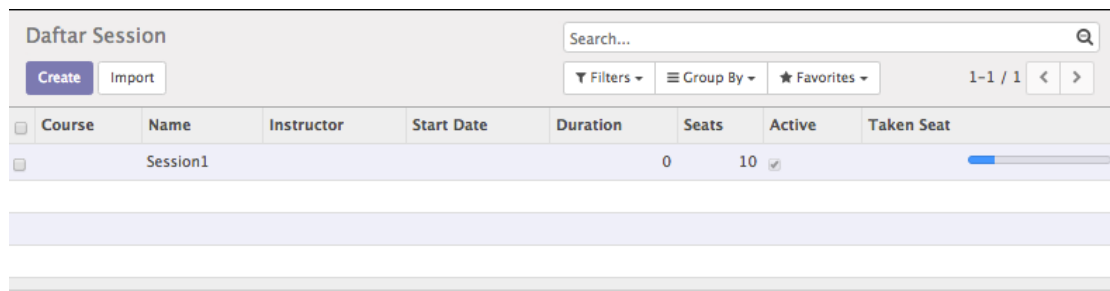
Progress bar dapat pula ditampilkan di tree view.

Edit `session.xml`, tambahi attribute `widget="progressbar"` pada field tambahan `taken_seats` di tree view.

```
<field name="arch" type="xml">
  <tree string="Session">
    <field name="course_id" />
    <field name="name" />
    <field name="instructor_id" />
    <field name="start_date" />
    <field name="duration" />
    <field name="seats" />
    <field name="active" />
    <field name="taken_seats" widget="progressbar"/>
  </tree>
</field>
```

Gambar 27 Nampilin `taken_seats` di tree dengan progress bar

Restart odoo dan update module, hasilnya...



The screenshot shows the 'Daftar Session' (Session List) interface in Odoo. It includes a search bar, filters, and a table with columns: Course, Name, Instructor, Start Date, Duration, Seats, Active, and Taken Seat. The 'Taken Seat' column displays a progress bar for 'Session1', indicating 0 out of 10 seats taken.

Course	Name	Instructor	Start Date	Duration	Seats	Active	Taken Seat
<input type="checkbox"/>	Session1				0	10	<input checked="" type="checkbox"/>

Gambar 28 Muncul progressbar persentasi seats di tree

Field `taken_seats` sudah muncul di tree view sebagai progress bar.

### 3 EVENT ONCHANGE

Sekarang kita mau agar nilai prosentasi taken\_seat berubah saat kita update nilai fields seats tanpa terlebih dahulu men-klik tombol Save.

Untuk ini kita perlu aktifkan event `on_change` pada field `seats`.

Untuk aktivasi event kita perlu lakukan step berikut:

- tambahi API decoration onchange dan
- definisikan function handler

Lanjut, edit file `session.py`, definisikan function event handler `onchange_seats()` di class Session.

```
@api.onchange('seats')
def onchange_seats(self):
    if self.seats>0:
        self.taken_seats = 100.0 * len(self.attendee_ids)/self.seats
    else:
        self.taken_seats = 0.0
```

*Gambar 29 Definisi function onchange event handler*

Sesuai ketentuan odoo, setiap event handler function mendapat parameter `self` yang berupa 1 object record yang sedang dibuka di form.

Ketika ada perubahan pada field yang ditentukan pada decorator `@api.onchange` maka function dibawahnya akan tereksekusi.

Untuk mengisi nilai field lain berdasarkan field yang berubah, tinggal diset saja nilai field tersebut dengan rumus yang dikehendaki melalui object record `self`.

PR: Gabungkan kedua logic rumus di onchange dan compute fields diatas.



## 4 CONSTRAINTS

Untuk menghindari kesalahan dalam penyimpanan data di database, kita bisa pake yang namanya constraints pada odoo.

Ada dua jenis constraints yang tersedia yaitu Python dan SQL constraints.

### 4.1 PYTHON CONSTRAINTS

Sebagai contoh kita buat constraints bahwa kalau seorang Partner udah menjadi instruktur dari suatu Session, dia nggak boleh lagi menjadi Attendee.

Buka file `session.py` dan tambah property `_constraints`.

```
@api.multi
def _cek_instruktur(self):
    for session in self:
        x = [att.partner_id.id for att in session.attendee_ids]
        if session.instructor_id.id in x:
            return False

    return True

_constraints = [(_cek_instruktur, 'Instructor cannot be Attendee',
                ['instructor_id', 'attendee_ids'])]
```

*Gambar 30 Cara menambah constraints*

Disini kita definisikan property `_constraints` yang berupa array list berisi tuple yang terdiri dari:

- fungsi pengecekan,
- warning message jika terjadi constraints,
- dan list dari fields yang mau dicek.

Prinsipnya dalam Python constraints, pengecekan dilakukan dalam sebuah function. Disini kita buat function `_check_instructor` yang dideklarasikan sebelum dia dipanggil dari property `_constraints`.

Function pengecekan constraint punya return value True atau False. Jika terjadi constraints maka return False. Tapi jika tidak terjadi apa-apa maka return True.

Dalam function `_check_instructor`, terdapat parameter `self` yang berupa record-record session yang akan dicek constraint-nya.

Lalu kita looping satu per satu record session yang ada pada variabel `self`, dan simpan ke local variable `session` seperti ini.

```
for session in self:
    x = [att.partner_id.id for att in session.attendee_ids]
    if session.instructor_id.id in x:
        return False
    return True
```

Dalam setiap loop kita jalankan list comprehension (fitur Python):

```
x = [att.partner_id.id for att in session.attendee_ids]
```

... yang artinya, pada setiap record `attendee_ids` yang ditemukan pada session (`session.attendee_ids`), simpan kedalam variable local `att`, lalu ambil field `partner_id.id` dari record attendee `att`, dan bentuk array list kedalam variable local `x`.

Pada akhirnya, `x` akan berisi array of partner id dari attendee yang hadir dalam session yang sedang dicari. Misalnya

```
x = [1,2,3,5]
```

lalu kita lakukan cek apakah partner id instruktur session tersebut ada di dalam array `x`.

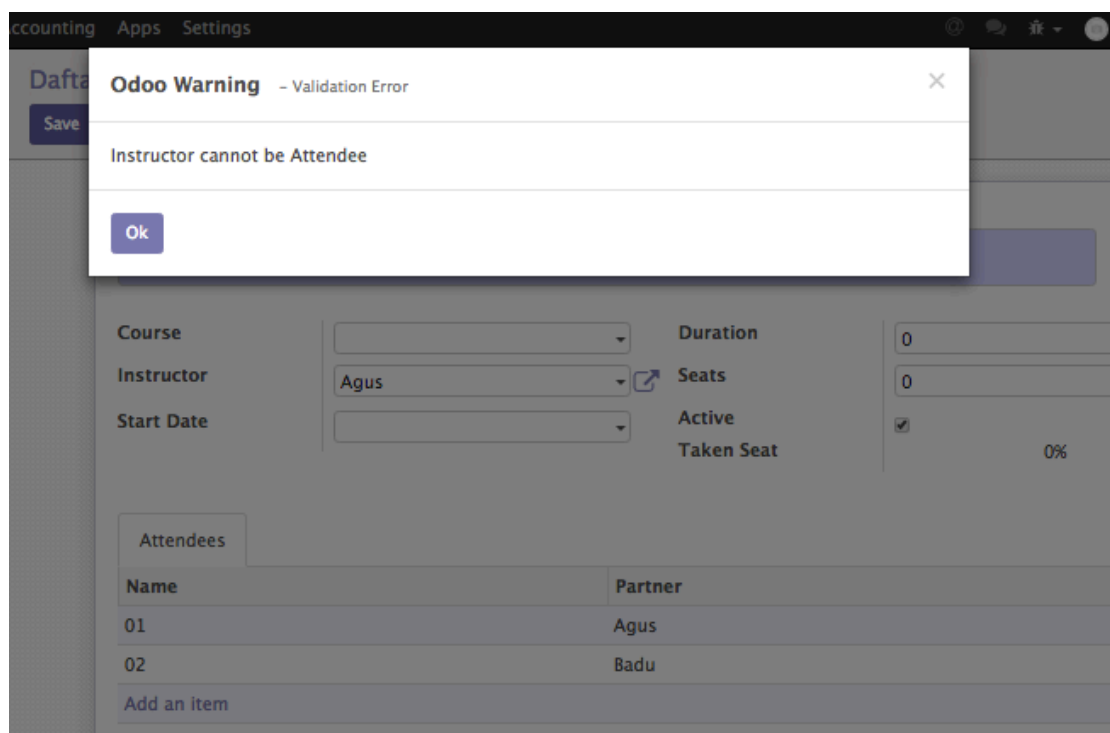
```
if session.instructor_id.id in x:
    return False
```

jika ada, langsung return False artinya, constraint nggak lolos.

Tapi kalau nggak ada instruktur pada array x maka constraint lolos, function return True.

Restart odoo dan update modul. Jadikan seorang partner sebagai instruktur sekaligus attendee.

Hasilnya



Gambar 31 Muncul warning kalau ada data tidak sesuai constraints

## 4.2 SQL CONTRAINS

Fungsinya sama seperti Python constraints yaitu untuk memprotek data jika ada batasan yang nggak lolos supaya nggak tersimpan ke database.

Namum syntaxnya menggunakan SQL constraints sesuai dokumentasi PostgreSQL

<http://www.postgresql.org/docs/8.4/static/ddl-constraints.html>.

Edit file `course.py`.

Tambahi SQL constraint pada **Course** class untuk:

- Membatasi agar description dan title Course nggak sama.
- Memastikan nama Course nggak ada yang double

```
from odoo import api, fields, models, _

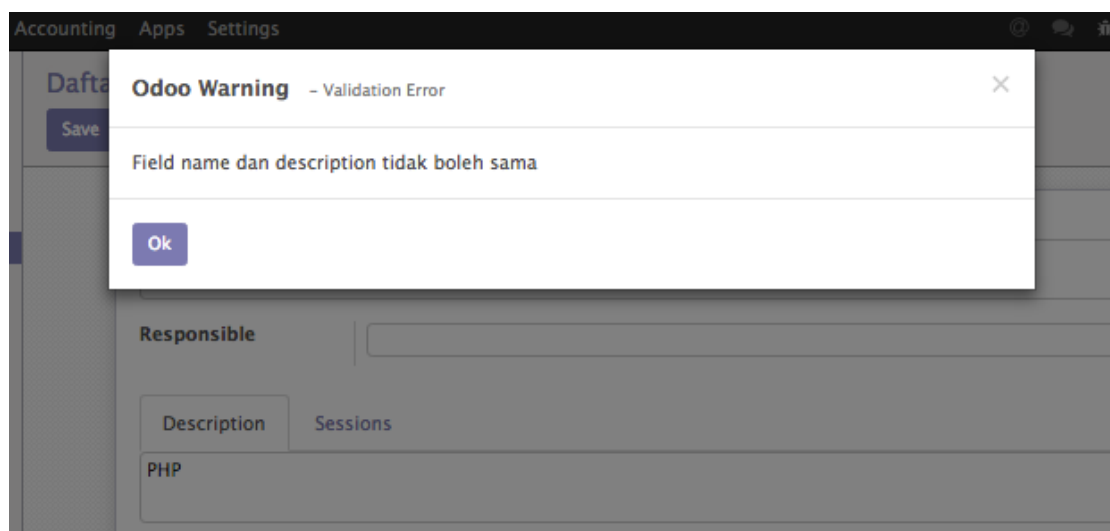
class Course(models.Model):
    _name = 'academic.course'
    _rec_name = 'name'

    name = fields.Char("Name")
    description = fields.Text(string="Description", required=False, )
    responsible_id = fields.Many2one(comodel_name="res.users",
                                    string="Responsible")
    session_ids = fields.One2many(comodel_name="academic.session",
                                  inverse_name="course_id",
                                  string="Sessions", required=False,
                                  ondelete="cascade")

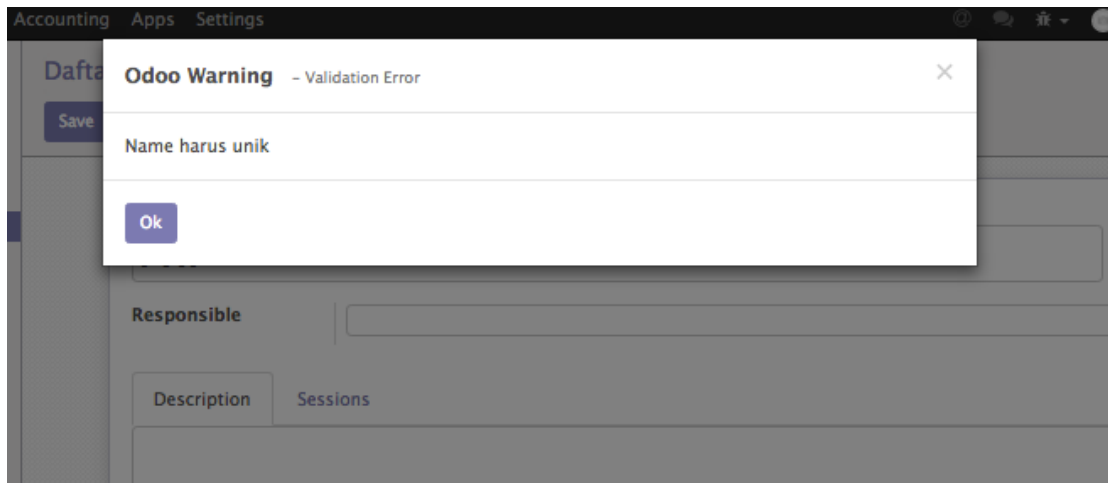
    _sql_constraints = [
        ('cek_name_desc', 'CHECK(name <> description)',
         'Field name dan description tidak boleh sama'),
        ('cek_unik_name', 'UNIQUE(name)',
         'Name harus unik')
    ]
```

Gambar 32 Cara nambahin SQL constraints di Session class

Restart odoo dan upgrade modul. Hasilnya, jika nama dan description Course sama.



Dan jika ada 2 nama Course yang sama:



Lanjut, edit `attendee.py`

Tambahi constraints agar pada table Attendee nggak bisa ada partner yang double pada satu Session yang sama.

```
from odoo import api, fields, models, _

class Attendee(models.Model):
    _name = 'academic.attendee'
    _rec_name = 'name'

    name = fields.Char("Name")
    session_id = fields.Many2one(comodel_name="academic.session",
                                string="Session", required=False, )
    partner_id = fields.Many2one(comodel_name="res.partner",
                                string="Partner", required=False, )

    _sql_constraints = [
        ('partner_session_unique', 'UNIQUE(partner_id,session_id)',
         'You cannot insert the same attendee multiple times!'),
    ]
```

Gambar 33 Cara nambahin SQL constraints di Attendee class

Restart odoo dan update module. Hasilnya, jika ada 2 partner dijadikan attendee pada session yang sama:

Accounting Apps Settings

Daftar


Save

**Odoo Warning** - Validation Error

You cannot insert the same attendee multiple times!

Ok

Course  Duration

Instructor   Seats

Start Date

Active ☒

Taken Seat 0%

Attendees

Name	Partner
01	Agus
02	Badu
	Agus

Add an item

## 5 NILAI DEFAULT – LAMBDA FUNCTION

Di atas kita udah bisa dengan mudah input dan update data tanpa banyak coding.

Gimana supaya isi beberapa field udah ada isinya secara otomatis waktu baru di-create? Misalnya by default field `active = True` dan `date_start` = tanggal hari ini waktu create Session.

Hal ini bisa dilakukan dengan menambahkan property `defaults` pada class setiap field di object Session.

Contoh seperti ini...

```
from odoo import api, fields, models, _
import time

class session(models.Model):
    _name = 'academic.session'

    name = fields.Char("Name", required=True)

    course_id = fields.Many2one(comodel_name="academic.course",
                                string="Course", required=False, )
    instructor_id = fields.Many2one(comodel_name="res.partner",
                                    string="Instructor", required=False, )
    start_date = fields.Date(string="Start Date", required=False,
                             default=lambda self:time.strftime("%Y-%m-%d"))
    duration = fields.Integer(string="Duration", required=False, )
    seats = fields.Integer(string="Seats", required=False, )
    active = fields.Boolean(string="Active", default=True)

    attendee_ids = fields.One2many(comodel_name="academic.attendee",
                                   inverse_name="session_id",
                                   string="Attendees",
                                   required=False, )
```

*Gambar 34 Default value fields*

Disini kita isi nilai default `active` dengan `True` dan `start_date` dengan lambda function yang mengeluarkan tanggal hari ini. Berhubung kita panggil function `time.strftime()` untuk mengetahui current date, maka kita perlu import module `time` Python di dalam source code Session.

Restart odoo dan update module.

Hasilnya ...

Gambar 35 Pada saat create otomatis ada nilai default

Disini untuk mengisi nilai default **start\_date** kita gunakan lambda function, yaitu function tanpa nama.

```
start_date = fields.Date(string="Start Date", required=False,
                        default=lambda self:time.strftime("%Y-%m-%d"))
```

Artinya, kita mengisi kolom **start\_date** dengan nilai yang dikeluarkan oleh lambda function, yaitu tanggal hari ini, yang dihasilkan oleh method **strftime()** dari package **time**, dengan format tahun-bulan-hari.

Lambda function di atas punya parameter function \*a. Di Python ini artinya bahwa function ini boleh punya lebih dari satu parameter. Jadi waktu digunakan, parameter boleh diisi dengan misalnya **f(1)**, **f(1,2,3)**, dan seterusnya.

Package Python **time** harus diimport terlebih dahulu pada class **Session**.

```
import time
```



## 5.1 APA ITU LAMBDA FUNCTION?

Di bahasa Python kita bisa bikin function tanpa nama (anonymous functions) saat runtime, menggunakan syntax construct yang namanya "lambda".

Contoh code di bawah ini menunjukkan perbedaan antara normal function ("f") dan lambda function ("g"):

```
>>> def f (x): return x**2
>>> print f(8)
64
>>>
>>> g = lambda x: x**2
>>>
>>> print g(8)
64
```

Seperti terlihat di atas, logika dan hasil function `f()` dan `g()` persis sama, tapi `g()` bisa dipanggil langsung tanpa perlu membuat definisi function terlebih dahulu seperti `def f()`.

## 5.2 PARAMETER FUNCTION \*X ?

Parameter function dalam Python bisa didefinisikan dengan `*x`, maksudnya function ini bisa dipanggil dengan jumlah parameter yang berbeda-beda.

Misalnya didefinisikan function seperti ini:

```
def foo(*args):
    for a in args
        print a
```

Maka function `foo()` itu boleh dipanggil dengan `foo(1)`, `foo(1,2,3)` dan sebagainya.

Hasilnya seperti ini:

```
foo(1)
```

```
1
foo(1,2,3,4)
1
2
3
4
```

Ada lagi bentuk parameter `**x`, yang artinya parameter function adalah berupa dictionary.

Misalnya di definisikan function seperti ini:

```
def bar(**kwargs):
    for a in kwargs:
        print a, kwargs[a]
```

Cara pemanggilan dan hasilnya:

```
bar(name="one", age=27)
age 27
name one
```

Untuk lebih jelasnya bisa cek ke TKP :

<http://docs.python.org/dev/tutorial/controlflow.html#more-on-defining-functions>

## 6 FITUR DUPLICATE

Odoo menyediakan fitur untuk men-duplicate data. Pada form view bagian atas ada tombol **More**. Di bawahnya ada menu **Duplicate**. By default semua isi field record yang di-duplicate sama persis dengan record baru hasil duplicate.

Kita bisa modif supaya record baru punya nilai field yang berbeda dengan aslinya.

Untuk ini kita harus override function `copy()` yang tersedia pada object bawaan odoo.

Modif file `session.py`, tambahi method `copy()` yang merupakan override dari parent-nya.

```
@api.multi
def copy(self, default=None):
    self.ensure_one()
    default = dict(default or {},
                    name=_('Copy of %s') % self.name)
    return super(session, self).copy(default=default)
```

*Gambar 36 Override method copy()*

Function `copy()` punya dua parameter bawaan. Salah satunya adalah `defaults` yang gunanya untuk nentuin nilai defaults dari field-field untuk record yang baru yang akan diinsert ke database.

Prinsipnya disini kita jalanin lagi function `copy()` yang ada pada parent class. Tapi pada saat pemanggilan itu, kita udah modif parameter `defaults` untuk record yang baru.

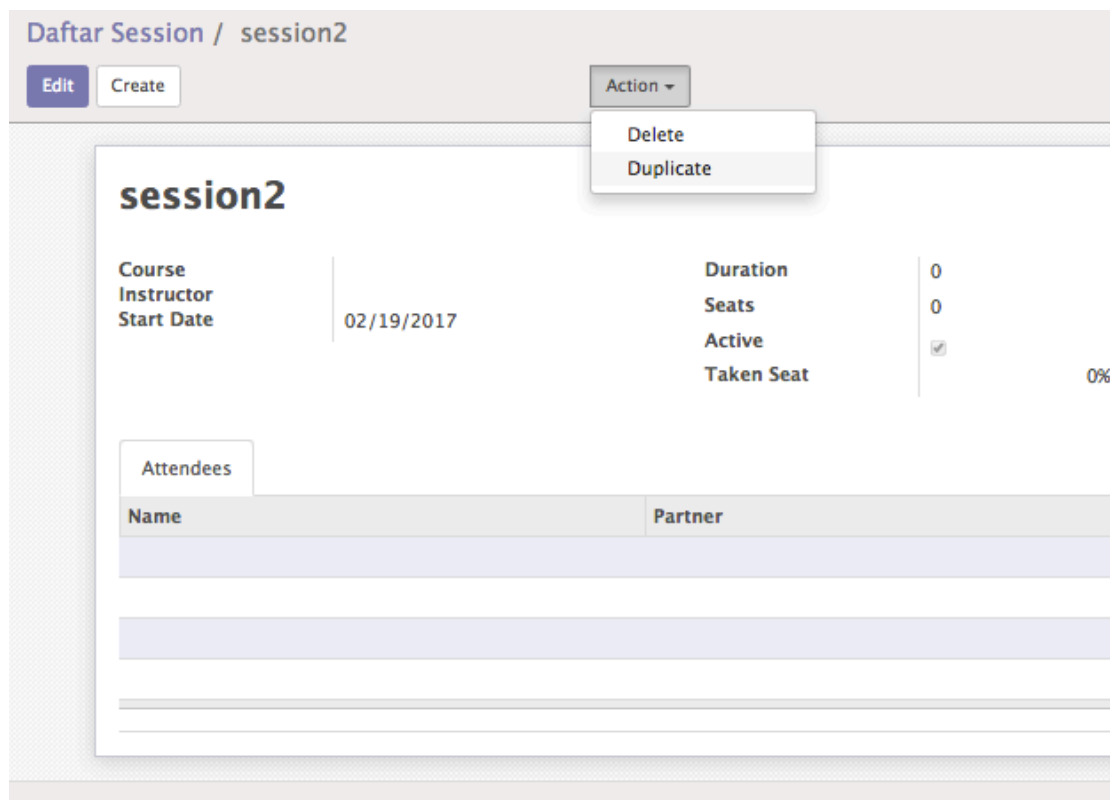
Dalam hal ini nilai `defaults` yang kita modif adalah untuk field `name`, yang kita ganti menjadi nama record sebelumnya tapi diawali dengan "Copy of "

```
default = dict(default or {},
                name=_('Copy of %s') % self.name)
```

variabel self berisi object record session yang hendak di-duplicate jadi kita bisa dengan mudah mengambil nilai field name dari record tersebut dengan self.name.

Restart odoo dan update module.

Hasilnya ...



Gambar 37 Tombol Duplicate

Record hasil duplicate namanya udah berubah sesuai yang kita mau...

Accounting Apps Settings Administrator (aca)

Daftar Session / Copy of session2

Save Discard 3 / 3

Session Name

Copy of session2

Course

Instructor

Start Date

Duration

Seats

Active

Taken Seat

0

0

☒

0%

Attendees

Name	Partner
Add an item	

Gambar 38 Nilai record yang diduplicate sudah berubah

## 7 REKAP HARI 2

Wow.... Banyak juga yang udah kita pelajari di Hari 2 ... berikut ini rekap nya..

Mengetahui class can view Inheritance – contoh class Instructor dan view form dan search nya.

Membuat Functional Fields – contohnya Percentage Taken Seats.

Mengambil event OnChange supaya taken seats berubah waktu seats diisi.

Membuat Constraints untuk membatasi Partner Instructor supaya nggak bentrok dengan Attendee.

Mengisi Nilai Default – menggunakan Lambda Function

Membuat Fitur Duplicate – meng-override function `copy()`.