1 HARI 3: ADVANCED VIEW

1.1 WARNA LIST VIEW

Baris-baris pada List tree view dapat dikasi warna sesuai dengan kondisi tertentu. Misalnya jika status Draft warnanya biru, kalau Rejected warna merah, dan sebagainya.

Disini kita coba bikin warna pada daftar Session, dimana jika `taken_seats` lebih dari 50% maka warna merah, namun jika masih dibawah 20% warna hijau.

Buka file `session.xml`.

Edit bagian session list view. Tambahi atribut `colors` pada tag `tree` seperti ini…

```
<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"
                 colors="red: taken_seats&gt;50; green:taken_seats&lt;20">
                    <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>
        </record>
```

Gambar 1 Nge-set warna baris tree view

Tag `colors` diatas isinya punya format seperti berikut:

```
warna: kondisi ; warna: kondisi; dst…
```

Perlu dicatat disini untuk kondisi lebih besar dan lebih kecil dari perlu dipakai HTML entities `>` dan `<` karena kita berada

di dalam XML, sehingga kalau menggunakan tanda > dan < akan menyebabkan error bentrok dengan tag XML < dan >.

Contohnya pada coding di atas:

```
red: taken_seats&gt;50; green:taken_seats&lt;20
```

artinya :

- jika `taken_seats` > 50 maka warna merah
- jika `taken_seats` < 20 maka warna hijau

Restart server dan update module.

Hasilnya…

Name	Course	Instructor	Start Date	Duration	Number of Seats	Is Active?	Attendees	Taken Seats
Session 1	Java	Joko	09/03/2014	0	20	☑	(7 records)	
Session 2	Java	Badu	09/04/2014	0	3	☑	(2 records)	
Session2	Java	Agus	09/02/2014	0	0	☑	(2 records)	
Session 4	Java	Ujang	09/02/2014	0	0	☑	(0 records)	
Copy of Session 1	Java	Joko	09/02/2014	0	5	☑	(3 records)	
Copy of Copy of Session 1	Java	Joko	09/02/2014	0	20	☑	(3 records)	
Copy of Session 4	Java	Ujang	09/02/2014	0	0	☑	(0 records)	
Session 1	PHP	Badu	09/03/2014	0	0	☑	(0 records)	

Gambar 2 Warna tree view udah berubah sesuai kondisi

1.2 Calendar View

odoo menyediakan tampilan data dalam bentuk Calendar selain dari tree dan form seperti yang udah sering kita pakai sebelumnya.

Disini kita coba untuk nampilin daftar Session dalam bentuk Calendar sesuai dengan tanggal `start_date` masing-masing session dan dibedakan warnanya berdasarkan Course-nya.

Buka file `session.xml`.

Tambahi record baru dengan `id` = `session_calendar` seperti ini...

```
<!-- calendar -->
<record id="session_cal" model="ir.ui.view">
   <field name="name">session_cal</field>
   <field name="model">academic.session</field>
   <field name="arch" type="xml">
      <calendar string="Session" date_start="start_date" color="course_id">
         <field name="name" />
      </calendar>
   </field>
</record>
```

Gambar 3 Bikin calendar view

Disini kita buat saru record baru untuk object `ir.ui.view` seperti pada waktu bikin tampilan tree dan form.

Field `name` diisi dengan identifikasi record ini misalnya `session.calendar`.

Field `model` diisi dengan nama object session yaitu `academic.session`.

Field `arch` isinya definisi tampilan calendar, yang menggunakan tag `calendar`.

Tag `calendar` punya atribut berikut:

- `string`, adalah label Calendar
- `date_start`, adalah nama field untuk tanggal dimulainya event pada Calendar, disini kita pakai field `start_date` pada object Session
- `color`, adalah nama field untuk pengelompokan warna, disini kita pakai field `course_id` pada object Session.

Lanjut, kita perlu edit action window list tree session supaya menampilkan icon tampilan calendar, selain dari form dan list view.

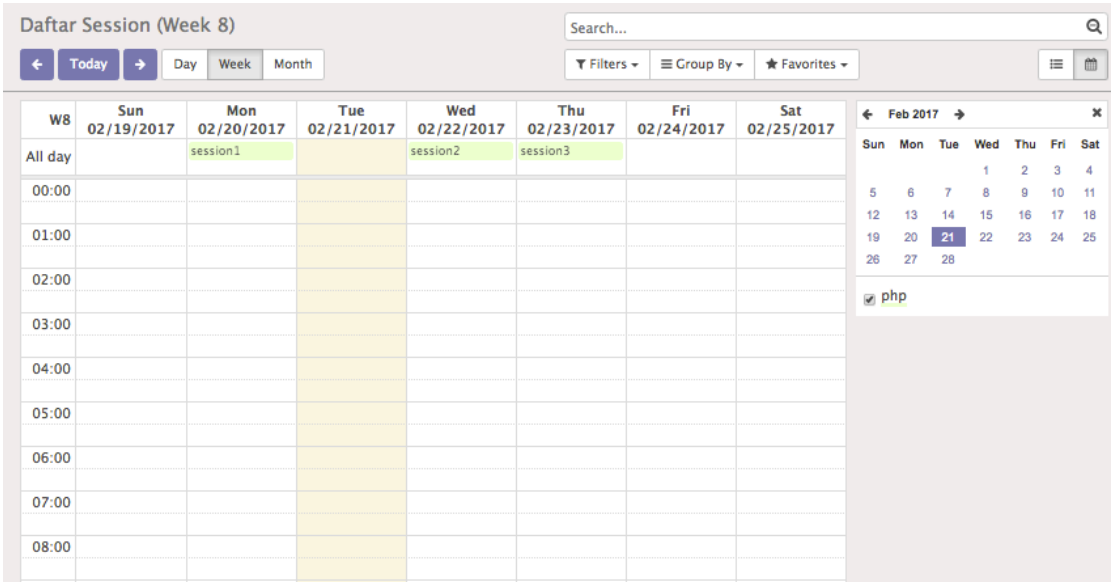
Buka file `menu.xml`.

Edit bagian action window session, dan tambahi calendar pada field `view_mode`.

```
<record id="action_session_list" model="ir.actions.act_window">
   <field name="name">Daftar Session</field>
   <field name="res_model">academic.session</field>
   <field name="view_mode">tree,form,calendar</field>
   <field name="help" type="html">
            <p class="oe_view_nocontent_create">
              Click to add a Session
            </p>
      <p>klik tombol create untuk bikin Session baru</p>
   </field>
</record>
```

Gambar 4 Aktivasi icon calendar view

Restart odoo dan update module, hasilnya...



Gambar 5 Session ditampilkan dalam calendar

1.3 Search View

Secara default odoo list view hanya menyediakan pencarian berdasarkan field name.

Kita bisa modif pencarian tersebut berdasarkan field-field lainnya sesuai kebutuhan.

Bahkan kita bisa kelompokkan record-record berdasarkan field secara bertingkat (grouping).

Sekarang kita coba untuk menambahkan pencarian berasarkan instruktur dan name.

Kita juga akan buat filtering untuk record yang memiliki durasi tidak sama dengan nol.

Lalu kita coba bikin grouping berdasarkan Course dan Start Date.

Buka file `session.xml`.

Lalu tambahin record baru untuk object `ir.ui.view` seperti ini...

```
<!-- search -->
<record id="session_search" model="ir.ui.view">

   <field name="name">session_search</field>
   <field name="model">academic.session</field>
   <field name="arch" type="xml">
      <search string="Search Session">

         <filter string="Non Zero Duration"
            name="non_zero"
            domain="[('duration','>',0)]" />

         <field name="name"/>
         <field name="instructor_id"/>

         <group string="Group By..">
            <filter string="Course" domain="[]"
               context="{'group_by':'course_id'}"/>

            <filter string="Instructor" domain="[]"
               context="{'group_by':'instructor_id'}"/>

            <filter string="Date" domain="[]"
               context="{'group_by':'start_date'}"/>
         </group>

      </search>
   </field>
</record>
```

Gambar 6 Definisi search view

Disini kita tambahi record baru untuk object `ir.ui.view`. Field name diisi session.filter.

Field model diisi dengan nama model session yaitu academic.session.

Field arch diisi dengan definisi search view menggunakan tag search.

Tag `search` isinya bisa berupa `filter`, `field`, atau `group`.

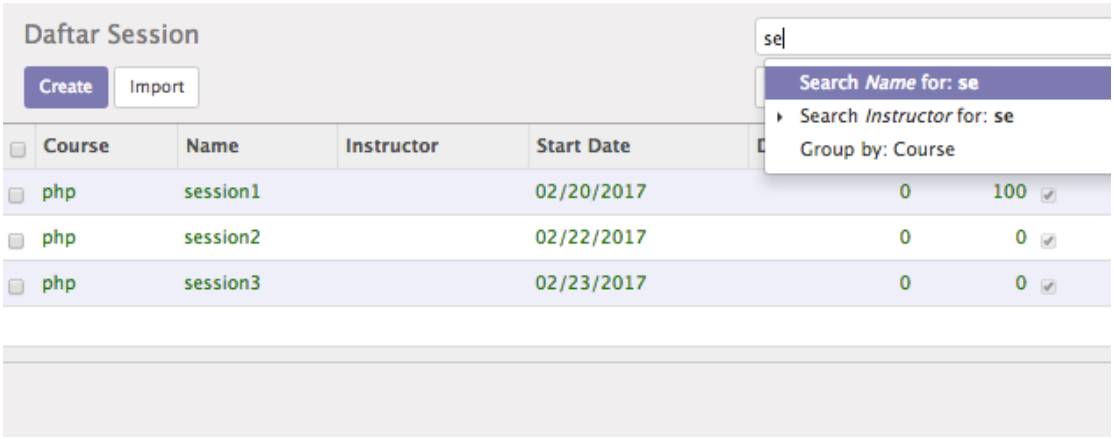
Tag `filter` gunanya untuk mem-filter data list view sesuai kriteria pada domain.

Tag `field` gunanya untuk supaya bisa mencari berdasarkan field tertentu.

Tag `group` gunanya untuk mengelompokkan list berdasarkan field tertentu. Didalam group buat lagi tag filter yang berkorelasi ke masing-masing field yang akan di-group. Set nama field untuk group pada atribut context.

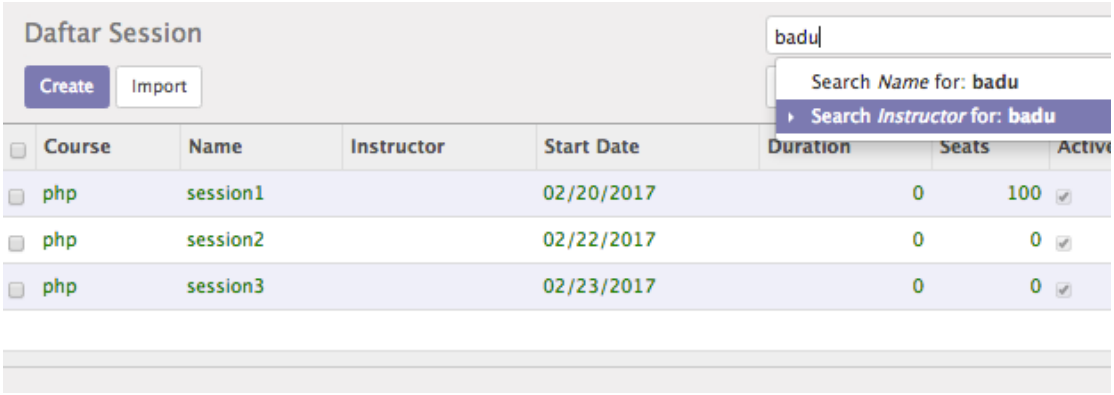
Restart odoo dan update module. Hasilnya…

Search view sudah lengkap sesuai yang kita tentukan sebelumnya, bisa search berdasarkan Session Name…



Gambar 7 Search dengan nama session

Bisa search berdasarkan nama instruktur…

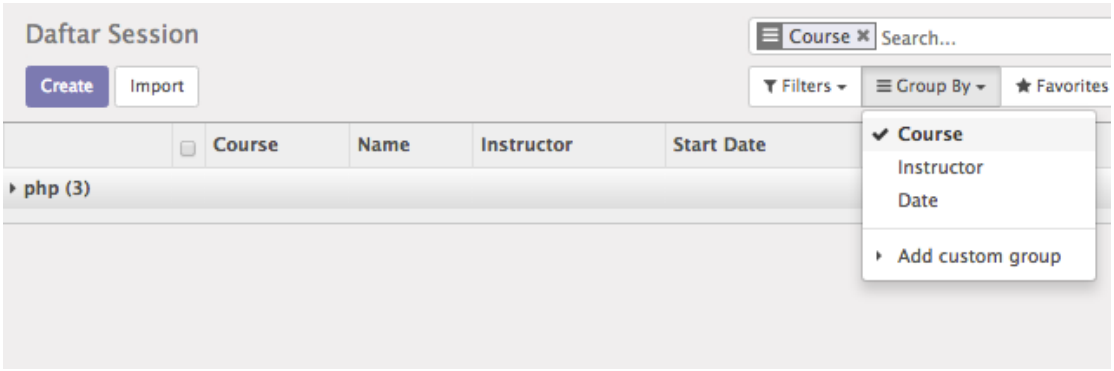


Gambar 8 Search dengan nama Instruktur

Bisa filter semua session yang durasinya nggak nol, yaitu melalui Non Zero Duration filter…

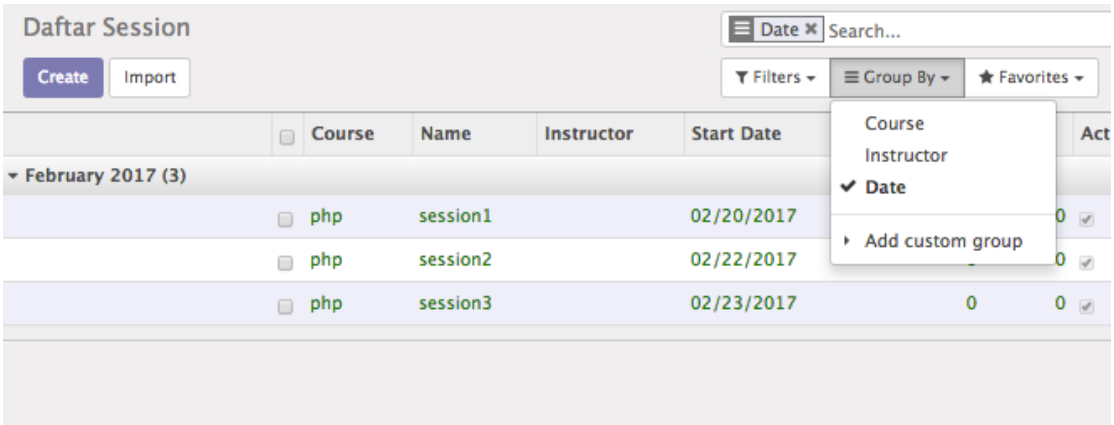
Gambar 9 Filter yang durasinya nggak nol

Bisa kelompokin Session berdasarkan Course-nya…



Gambar 10 Group berdasarkan Course

Dan bisa kelompokin session berdasarkan Start Date nya



Gambar 11 Group berdasarkan tanggal

1.4 GANTT VIEW

Catatan: Untuk Odoo10, gantt view sudah tidak ada di versi Community, dan hanya tersedia di versi Enterprise.

odoo juga menyediakan tampilan data dalam bentuk Gantt Chart, untuk memudahkan perencanaan jadwal suatu data.

Disini kita coba untuk nampilin Session dalam Gantt Chart.

Pertama-tama kita harus aktivkan icon gantt view pada action window session list.

Edit `menu.xml`

Tambahi gantt pada field `view_mode` action window session list.

```
<record id="action_session_list" model="ir.actions.act_window">
   <field name="name">Daftar Session</field>
   <field name="res_model">academic.session</field>
   <field name="view_mode">tree,form,calendar,gantt</field>
   <field name="help" type="html">
            <p class="oe_view_nocontent_create">
              Click to add a Session
            </p>
      <p>klik tombol create untuk bikin Session baru</p>
   </field>
</record>
```

Gambar 12 Aktivasi gantt view icon

Ini mengakibatkan icon gantt view muncul pada Session list view.

Edit `session.xml`, tambahi satu record baru object `ir.ui.view`.

```
<!-- gantt -->
<record id="session_gantt" model="ir.ui.view">
    <field name="name">session_gantt</field>
    <field name="model">academic.session</field>
    <field name="arch" type="xml">
        <gantt date_delay="duration" string="Session"
            date_start="start_date"
            default_group_by="course_id">
        </gantt>
    </field>
</record>
```

Gambar 13 Definisi gantt view

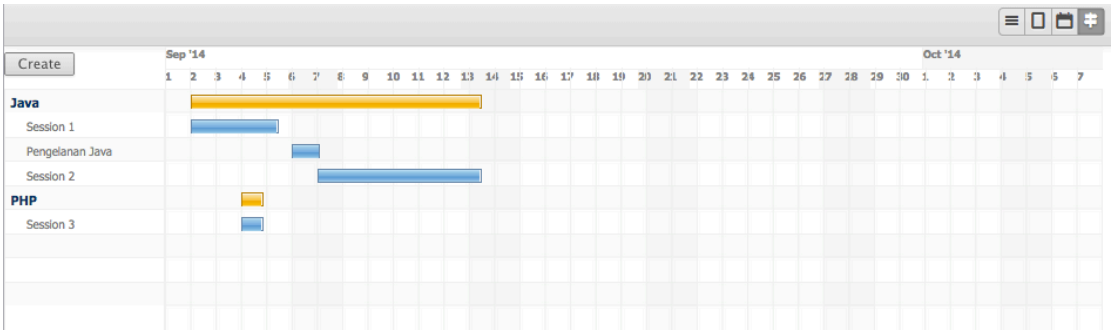
Sama seperti record yang lainnya, view ini punya field `name=session.gant`, dan field `model=academic.session`.

Field `arch` berisi definisi gantt chart yang dideklarasikan dengan tag `gantt`. Tag ini punya atribut sebagai berikut:

- `date_delay` menentukan berapa panjang bar pada gantt chart
- `date_start` menentukan tanggal mulai
- `string` menentukan label judul gantt chart
- `default_group_by` menentukan field yang dipakai untuk mengelompokkan gantt chart

Restart odoo dan update module.

Hasilnya…



Gambar 14 Session dalam gantt view per Course

1.5 Chart/ Graph View

odoo juga menyediakan tampilan data dalam bentuk grafik, untuk memudahkan analisa suatu data.

Disini kita coba untuk nampilin Session dalam grafik Bar dan Pie Chart.

Pertama-tama kita harus aktifkan icon graph view pada action window session list.

Edit `menu.xml`. Tambahkan `graph` pada field `view_mode`.

```
<record id="action_session_list" model="ir.actions.act_window">
   <field name="name">Daftar Session</field>
   <field name="res_model">academic.session</field>
   <field name="view_mode">tree,form,calendar,gantt,graph</field>
   <field name="help" type="html">
            <p class="oe_view_nocontent_create">
              Click to add a Session
            </p>
      <p>klik tombol create untuk bikin Session baru</p>
   </field>
</record>
```

Gambar 15 Aktivasi graph view icon

Ini mengakibatkan icon graph view muncul pada Session list view.

Edit `session.xml`. Tambahi satu record baru object `ir.ui.view`.

```
126
127         <record id="view_session_graph" model="ir.ui.view">
128             <field name="name">session.graph</field>
129             <field name="model">academic.session</field>
130             <field name="arch" type="xml">
131                 <graph string="Session" type="bar">
132                     <field name="instructor_id"/>
133                     <field name="seats" operator="+"/>
134                 </graph>
135             </field>
136         </record>
137
```

Gambar 16 Definisi graph view

Sama seperti record yang lainnya, view ini punya field `name=session.graph`, dan field `model=academic.session`.

Field `arch` berisi definisi graph yang dideklarasikan dengan tag `graph`. Tag ini punya atribut sebagai berikut:

- `string` , menentukan judul grafik
- `type`, menentukan type grapfik secara default, misalnya "`bar`"

- `orientation`, menentukan arah grafik apakah `horizontal` atau `vertical`

Di dalam tag `graph` ditentukan field yang dipakai sebagai data untuk menampilkan grafik.

Field pertama akan menjadi data sumbu X, field kedua menjadi sumbu Y, dan field ketiga jika ada menjadi sumbu Z.

Field kedua dan ketiga boleh ada tambahan atribut:

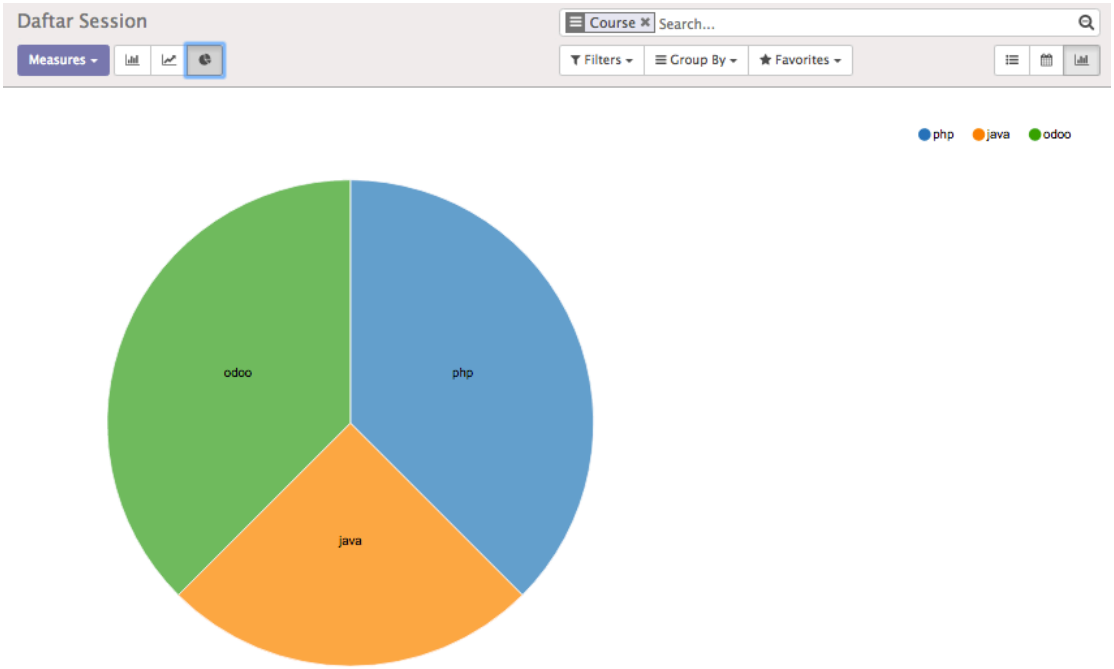
- `group`, yang menentukan field group by
- `operator`, yang gunanya menentukan operasi agregat yang digunakan untuk untuk field lainnya ketika salah satu field dibuat group by (pilihannya +,*,**,min,max)

Restart odoo dan update module.

Hasilnya…

Gambar 17 Graph berapa jumlah total seat per instruktur dalam bar chart

Pilih group by Course dan type pie chart…



Gambar 18 Graph berapa jumlah total seat per course dalam pie chart

1.6 KANBAN

odoo juga menyediakan tampilan data dalam bentuk icon/ kanban, untuk memudahkan visualisasi suatu data.

Pertama-tama kita harus aktifkan icon kanban view pada action window session list.

Edit `menu.xml`.

```
<record id="action_session_list" model="ir.actions.act_window">
   <field name="name">Daftar Session</field>
   <field name="res_model">academic.session</field>
   <field name="view_mode">tree,form,calendar,gantt,graph,kanban</field>
   <field name="help" type="html">
            <p class="oe_view_nocontent_create">
              Click to add a Session
            </p>
      <p>klik tombol create untuk bikin Session baru</p>
   </field>
</record>
```

Gambar 19 Aktivasi kanban view icon

Ini mengakibatkan icon graph view muncul pada Session list view.

Edit `session.xml`. Tambahi satu record baru object `ir.ui.view`.

```
<!-- kanban -->
<record id="session_kanban" model="ir.ui.view" >
    <field name="name">session_kanban</field>
    <field name="model">academic.session</field>
    <field name="arch" type="xml">
        <kanban version="7.0" default_group_by="course_id">
            <templates>
                <t t-name="kanban-box">
                    <div class="oe_kanban_vignette">
                        <a type="open">
                           <img
                            t-att-src="kanban_image('academic.session',
                            'image_small', record.id.value)"
                            class="o_kanban_image"/>
                        </a>

                        <div class="oe_kanban_details">
                            <h4>
                                <a type="open"><field name="name"/></a>
                            </h4>
                            <ul>
                                <li>
                                   Seats: <field name="seats"></field>
                                </li>
                                <li>
                                   Taken Seats:
                                   <field name="taken_seats"> </field>
                                </li>
                            </ul>
                        </div>
                    </div>
                </t>
            </templates>
        </kanban>
    </field>
</record>
```

Gambar 20 Definisi kanban view

Sama seperti record yang lainnya, view ini punya field `name=session.graph`, dan field `model=academic.session`.

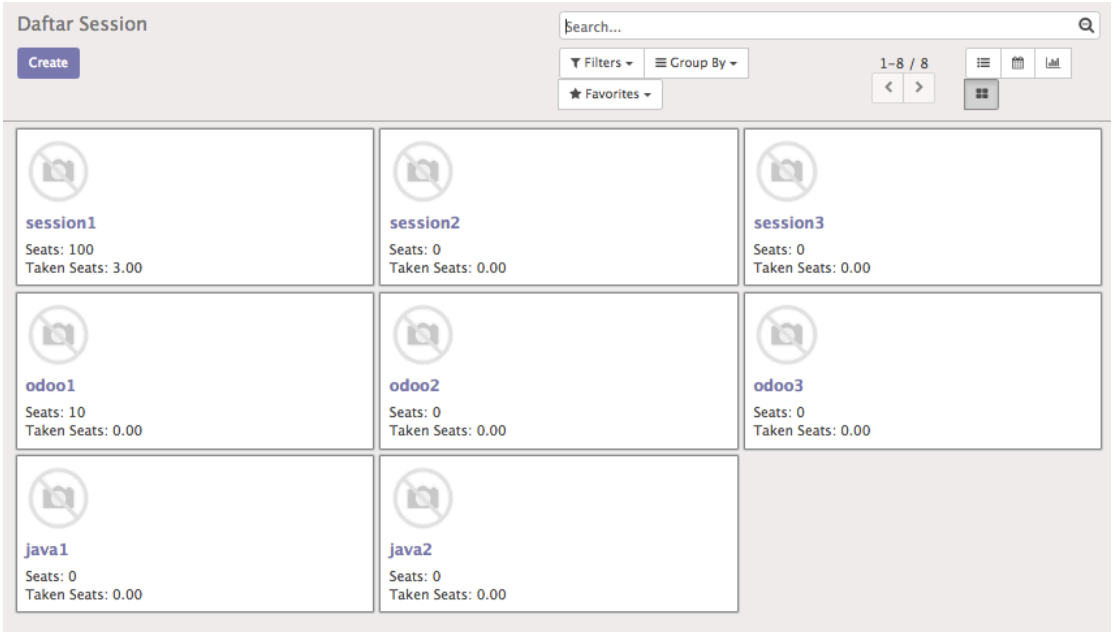
Field `arch` berisi definisi kanban yang dideklarasikan dengan tag `kanban`.

Di dalam tag `kanban` terdapat tag `template` yang merupakan template untuk menampilkan 1 record data.

Di dalam tag `template` kita boleh mix antara tag QWeb dan HTML. Contoh di atas, kita membuat tampilan icon Session sama seperti tampilan icon pada Customer di odoo, yaitu ada image, nama Session, Seats, dan Taken Seats.

Restart odoo dan update module.

Hasilnya…



Gambar 21 Session dalam kanban view

Tampilan kanban dapat dikelompokkan berdasarkan suatu field tertentu secara default sewaktu kanban itu dibuka.

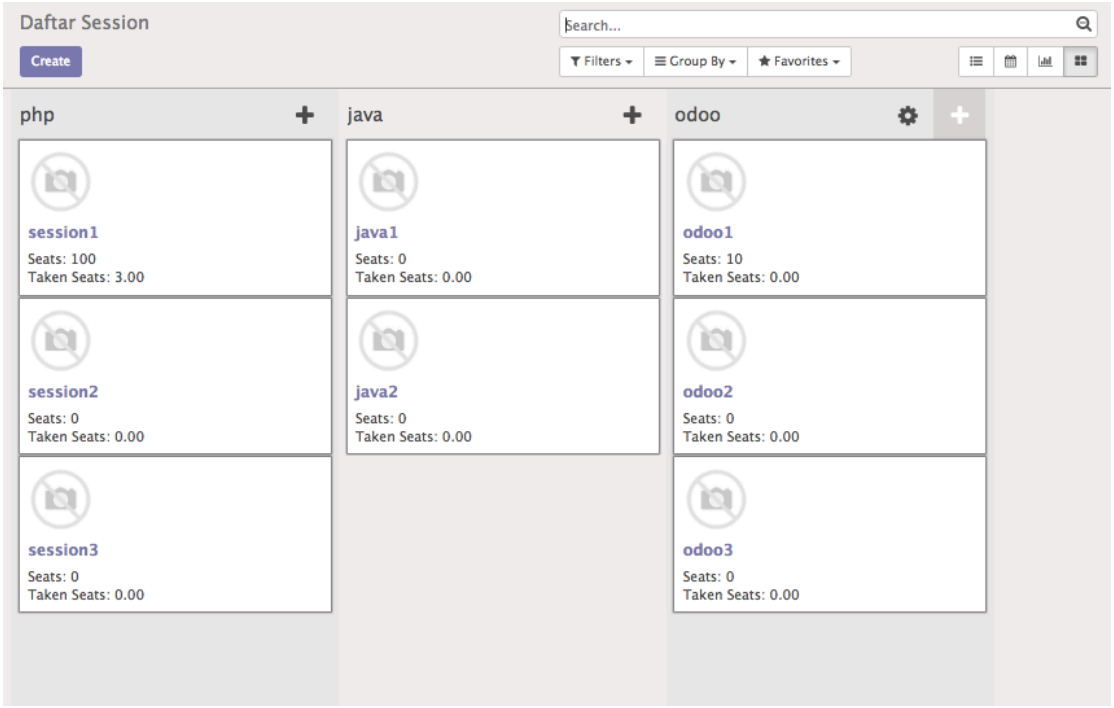
Contohnya kita mau kelompokin berdasarkan Course. Tambahi atribut default_group_by="course_id" pada tag kanban.

```
<!-- kanban -->
<record id="session_kanban" model="ir.ui.view" >
    <field name="name">session_kanban</field>
    <field name="model">academic.session</field>
    <field name="arch" type="xml">
        <kanban version="7.0" default_group_by="course_id">
            <templates>
                <t t-name="kanban-box">
```

Gambar 22 Session di-group per Course

Restart odoo dan update module.

Hasilnya...



Gambar 23 Kanban session per Course

1.7 Nambahin Field Image di Session

Pada deklarasi kanban sebelumnya, terdapat sebuah field pada template untuk nampilin gambar icon session, yaitu `image_small`. Pada kanban sekarang itu ditampilkan gambar kosong karena memang kita belum punya field `image_small` pada Session.

Sekarang kita tambahin, supaya pada kanban muncul gambarnya dan bisa diupload dari computer.

Edit file `session.py`, tambahi field baru namanya `image_small` dengan jenis `binary`.

```
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, )

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

    image_small  = fields.Binary(string="Image Small",  )
```

Gambar 24 Tamhai field image_small type binary

Lalu edit form view Session pada file `session.xml`.. tambahkan field `image_small` setelah `start_date`.

Field itu langsung kita kasi atribut `widget="image"` supaya pada form edit muncul image nya dan bisa kita ganti langsung disitu.

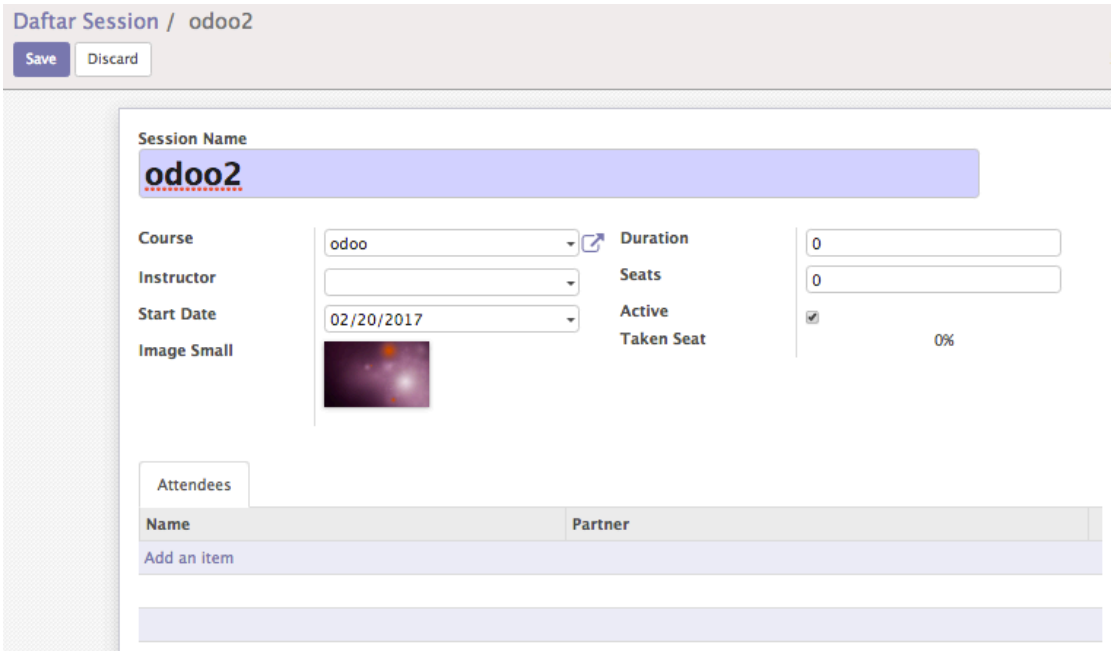
```
<record id="view_academic_session_form" model="ir.ui.view">
    <field name="name">academic.session.form</field>
    <field name="model">academic.session</field>
    <field name="type">form</field>
    <field name="priority" eval="8"/>
    <field name="arch" type="xml">
        <form string="Session">
            <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),
```

```
                                    ('category_id','=','Pelatih')]"/>
                        <field name="start_date" />
                        <field name="image_small"
                                    class="oe_left oe_avatar"/>
                    </group>
                    <group>
```

Gambar 25 Munculkan field image_small di form view Session

Restart odoo dan update module.

Hasilnya...



Gambar 26 Field image_small muncul sebagai image field, bisa upload image

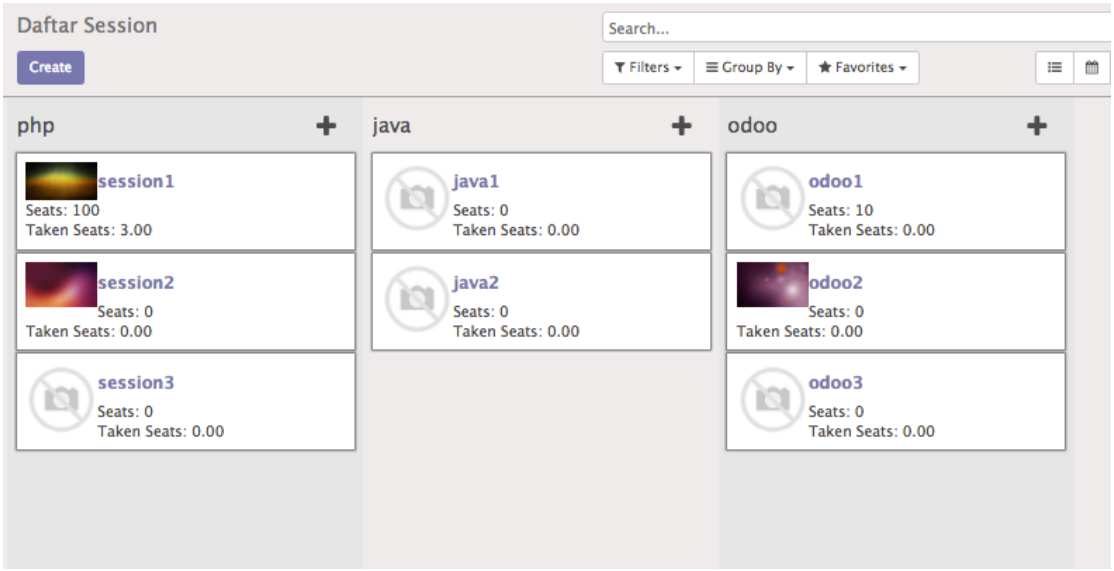
Cek lagi kode template kanban view pada file `session.xml`, tag `img` atribut `t-att-src` pada parameter function kanban_image() nama model dan field nya harus `academic.session` dan `image_small`.

```
<!-- kanban -->
<record id="session_kanban" model="ir.ui.view" >
    <field name="name">session_kanban</field>
    <field name="model">academic.session</field>
    <field name="arch" type="xml">
        <kanban version="7.0" default_group_by="course_id">
            <templates>
                <t t-name="kanban-box">
                    <div class="oe_kanban_vignette">
                        <a type="open">
                            <img
                            t-att-src="kanban_image('academic.session',
                            'image_small', record.id.value)"
                            class="o_kanban_image"/>
                        </a>
```

Gambar 27 Munculkan image_small pada kanban view

Restart odoo dan update module.

Hasil di kanban Session...



Gambar 28 Image muncul pada kanban

1.8 Related Field – Apa Nama Course suatu Attendee ?

Lanjut... gimana kalo kita mau tau apa nama **Course** dari record **Attendee**? Misalnya untuk keperluan pencarian dan grouping Attendee berdasarkan Course.

Caranya adalah dengan nambah kolom dengan type `related` pada file `attendee.py`.

Disitu kita definisiin gimana relasinya model Course:

```
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!'),
    ]

    course_id = fields.Many2one(comodel_name="academic.course",
                    string="Course",
                    required=False,
                    related="session_id.course_id",
                    store=True)
```

Gambar 29 Tambahin field related di Attendee class

Field related punya banyak parameter.

Parameter pertama nunjukin nama field yang ada pada class itu sendiri, yaitu `session_id`, yang mengandung informasi yang akan kita cari (nama Course).

Paremeter kedua adalah nama field (`course_id`) yang ada pada class parameter sebelumnya (`session_id`) yang mengandung informasi yang akan kita cari.

Demikian seterusnya sampai informasi yang kita mau cari udah tersedia pada class yang diwakili oleh field tersebut. Disini kita berhenti pada field `course_id` karena pada class Course kita udah nemu apa yang mau dicari yaitu nama Course.

Jika udah ketemu baru diset parameter berikutnya, yaitu `type="many2one"` , yang menunjukkan jenis relasi.

Parameter `relation` berisi nama class target yang mengadung informasi yang kita cari, dalam hal ini `academic.course`.

Parameter `store` berisi `True` atau `False` menandakan apakah kita akan simpan field ini ke table database. Kalau untuk grouping field ini harus disimpan.

Setelah itu, kolom `course_id` dapat dipakai untuk filter dan grouping seperti kolom regular lainnya.

Edit file `attendee.py` , tambahin satu record `ir.ui.view` baru untuk search view model `academic.attendee`.

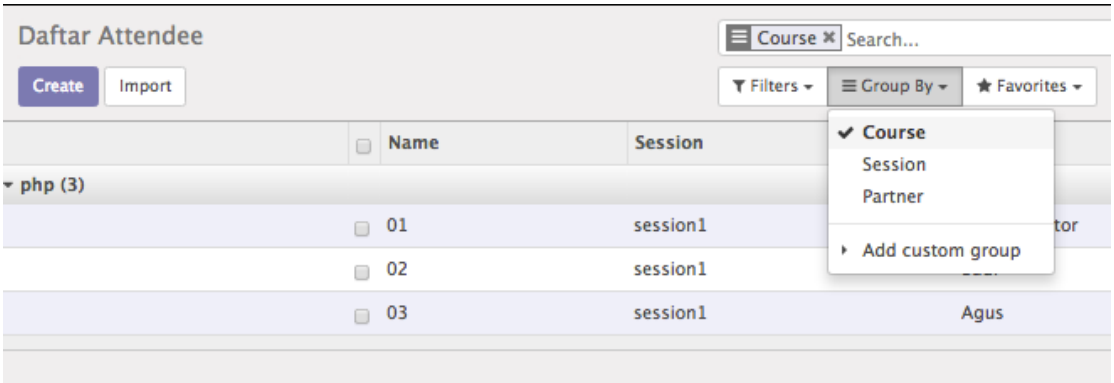
```
<!-- search -->
<record id="attendee_search" model="ir.ui.view">
   <field name="name">attendee_search</field>
   <field name="model">academic.attendee</field>
   <field name="arch" type="xml">
      <search string="Search Attendees">
               <field name="session_id"/>
               <field name="name"/>
               <group expand="1" string="Group By...">
                   <filter string="Course"
                       icon="terp-personal" domain="[]"
                        context="{'group_by':'course_id'}"/>
                   <filter string="Session" icon="terp-personal"
                        domain="[]"
                        context="{'group_by':'session_id'}"/>
                   <filter string="Partner"
                       icon="terp-personal" domain="[]"
                        context="{'group_by':'partner_id'}"/>

               </group>
           </search>
   </field>
</record>
```

Gambar 30 Bikin filter untuk Attendee class

Disini kita bisa munculkan pada grouping field `course_id` sama seperti field lainnya, misalnya `session_id` dan `partner_id`.

Restart odoo dan update module.

Hasilnya...



Gambar 31 Attendee bisa di-group berdasarkan Course

2 WORKFLOW

Workflow gunanya untuk mengupdate status dari suatu object sesuai kewenangan dari masing-masing bagian sesuai standard operating procedure.

Pada odoo kita bisa implementasikan workflow secara statis maupun dinamis.

Pada workflow statis, proses alur perpindahan status diimplementasikan secara hardcoded, yaitu langsung pada coding. Jika terjadi perubahan alur, maka harus dilakukan pada coding.

Sementara pada workflow dinamis, proses alur digambarkan pada workflow diagram. Jika terjadi perubahan alur maka cukup dilakukan pada diagram.

2.1 WORKFLOW STATIS

Pertama, kita bikin dulu field `state` yang nantinya dipakai untuk menentukan workflow pada object "Session".

Disini contohnya, Session boleh punya 3 states: Draft (default), Confirmed dan Done. Informasi state ini perlu ditampilkan pada view form session tapi readonly.

Untuk mengubah state dilakukan dengan click tombol-tombol operasional (Confirm, Mark as Done, dan Reset to Draft).

Transisi antar state yang valid:

- Draft → Confirmed
- Confirmed → Draft
- Confirmed → Done
- Done → Draft

Edit file `session.py`.

Tambahi dulu variable global `SESSION_STATE` yang isinya list array dari state-state yang ada pada Session.

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

SESSION_STATES =[('draft','Draft'),('confirmed','Confirmed'),
                 ('done','Done')]

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

Gambar 32 Definisi global variabel SESSION_STATES

Lalu , tambahi field `state` di Session class. Tipe nya `selection` dengan mengambil nilai pilihan dari variable global `SESSION_STATES`. Sifatnya readonly dan nggak boleh kosong.

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

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

    ...

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

    image_small  = fields.Binary(string="Image Small",  )

    state = fields.Selection(string="State", selection=SESSION_STATES,
                             required=True,
                             readonly=True,
                             default=SESSION_STATES[0][0])
```

Gambar 33 Tambah kolom state

Tambahi default value untuk field `state`, yaitu `draft`, `SESSION_STATES[0][0]`. Artinya elemen pertama dari elemen pertama array `SESSION_STATES`.

Definisikan 3 method baru di Session class. Method-method ini gunanya untuk mengupdate state dari Session, dipanggil waktu tombol-tombol workflow yang terkait di-click.

Waktu `action_confirm` dipanggil, maka `state` Session berubah menjadi Confirmed.

Waktu `action_done` dipanggil, maka `state` Session berubah menjadi Done.

Waktu `action_draft` dipanggil, maka `state` Session berubah menjadi Draft.

```
state = fields.Selection(string="State", selection=SESSION_STATES,
                         required=True,
                         readonly=True,
                         default=SESSION_STATES[0][0])


@api.multi
def action_draft(self):
    self.state = SESSION_STATES[0][0]

@api.multi
def action_confirm(self):
    self.state = SESSION_STATES[1][0]

@api.multi
def action_done(self):
    self.state = SESSION_STATES[2][0]
```

Gambar 34 Method action workflow

Lanjut... edit file `session.xml`.

Bikin 3 buttons di form view Session, di bagian “header” di atasnya “sheet”. Masing-masing button memanggil method yang terkait.

```
   <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">
               <header>
                   <button string="Confirm" type="object"
```

```
                        name="action_confirm"
                        states="draft" />
                    <button string="Mark as Done" type="object"
                        name="action_done"
                        states="confirmed" />
                    <button string="Reset to Draft" type="object"
                        name="action_draft"
                        states="confirmed,done" />
                    <field name="state" widget="statusbar" />
                </header>
```

Gambar 35 Definisi tombol-tombol workflow

Tombol-tombol ini ber-type object, artinya langsung terkait dengan object class Session. Atribut name langsung berarti nama method yang akan dijalankan waktu diklik yang ada pada object Session.

Atribut `states` menentukan kemunculan tombol ini sesuai `state` nya.

Tombol `Confirm` akan memanggil method `action_confirm`, dan muncul hanya ketika state masih `Draft`.

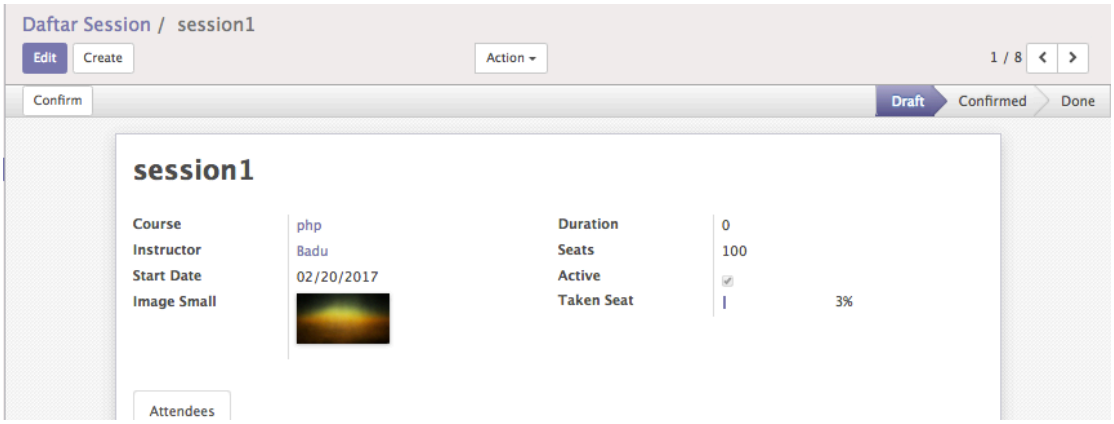
Tombol `Mark as Done` akan memanggil method `action_done`, dan muncul hanya ketika state sudah `Confirmed` .

Tombol `Reset to Draft` akan memanggil method `action_draft`, dan muncul hanya ketika state sudah `Confirmed` atau `Done`.

Tambahi juga field `state` dengan widget `statusbar` yang isinya diambil dari field state.

Restart odoo dan update module.

Hasilnya…



Gambar 36 Workflow static udah bisa jalan

Terdapat tombol `Confirm`. Waktu diclick, state berubah menjadi `Confirmed` dan muncul tombol `Mark as Done` dan `Reset to Draft`.

Waktu `Mark as Done` diclick, state berubah menjadi `Done`, tapi kalau `Reset to Draft` yang diclick, state balik menjadi `Draft`.

Setelah `Done`, muncul tombol `Reset to Draft`. Kalau diclick state balik menjadi `Draft`.

Workflow di atas udah ok, tapi sifat nya static. Kalau ada perubahan SOP maka harus ubah coding lagi.... ☹

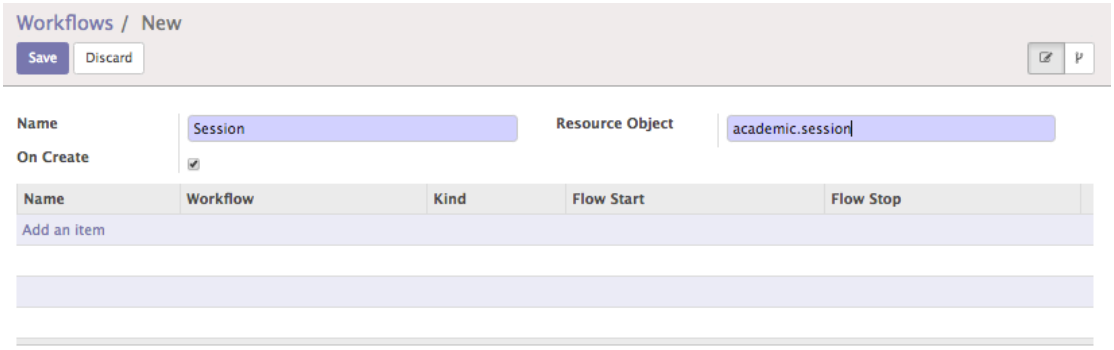
2.2 Dynamic Workflow – Bikin Diagram Workflow

Untung ada dynamic workflow ..... ☺

Lanjut, kita bikin workflow sama seperti di atas tapi sekarang pake workflow editor.

Klik menu `Settings > Technical > Workflows > Workflows`

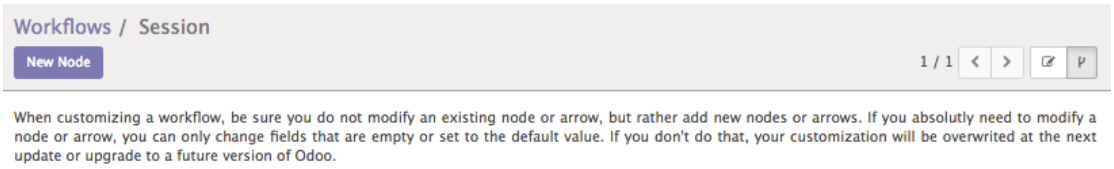
dan **Create** workflow baru.



Gambar 37 Membuat diagram workflow

Masuk ke diagram view dan tambahi nodes dan transitions.

- Panah transition harus berasosiasi dengan signal,
- Setiap Activity (= node) harus memanggi function yang memodifikasi session state, sesuai state pada workflow.



Gambar 38 Diagram view

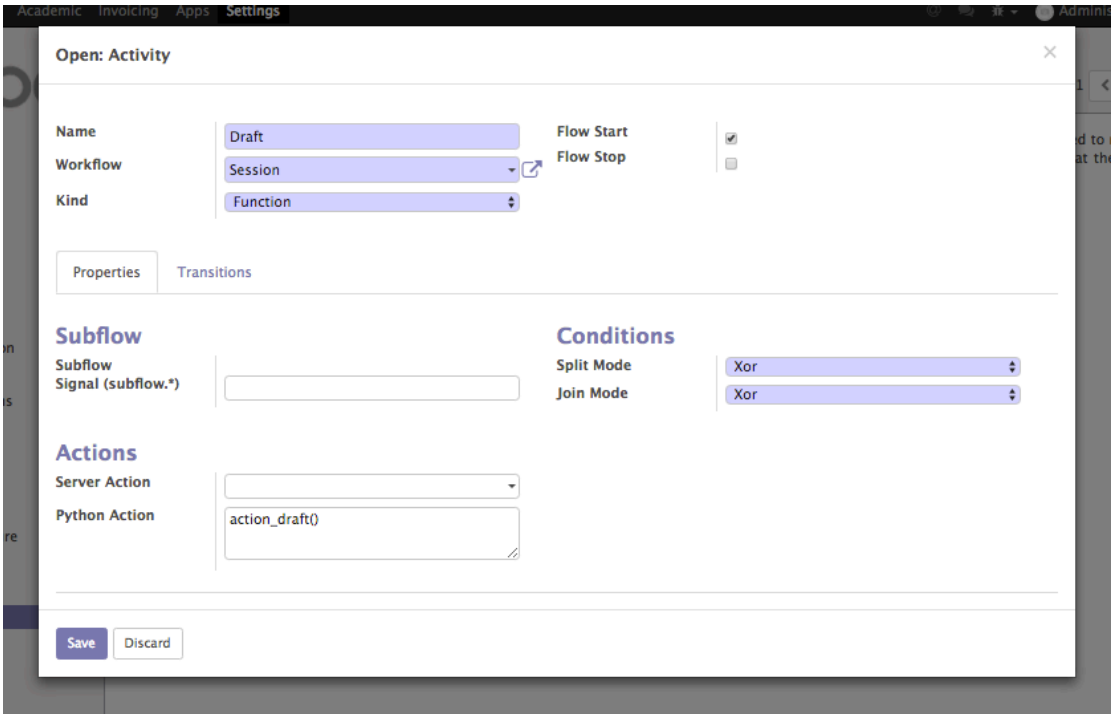
Klik **New Node:**

Name = Draft

Kind = Function

Python Action = action_draft()

Klik **Save**.



Gambar 39 Tambah node activity Draft

Klik **New Node** lagi:

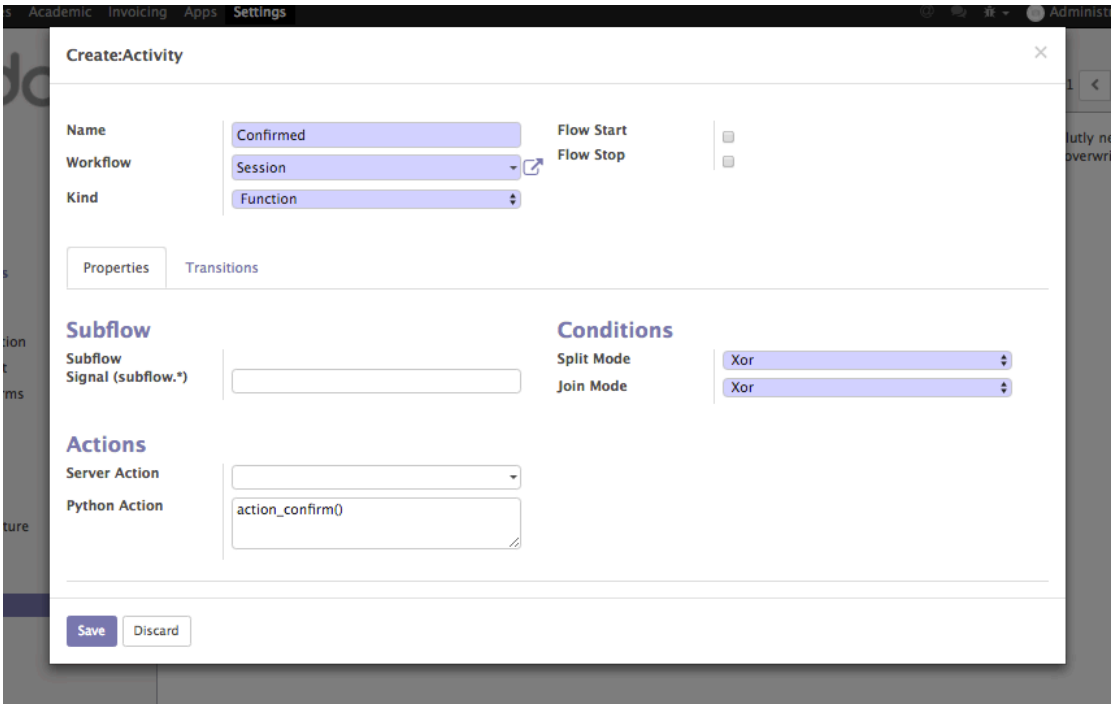
Name = Confirmed

Kind = Function

Python Action = action_confirm()

Flow Start = ya

Klik **Save**.



Gambar 40 Tambah node activity Confirmed

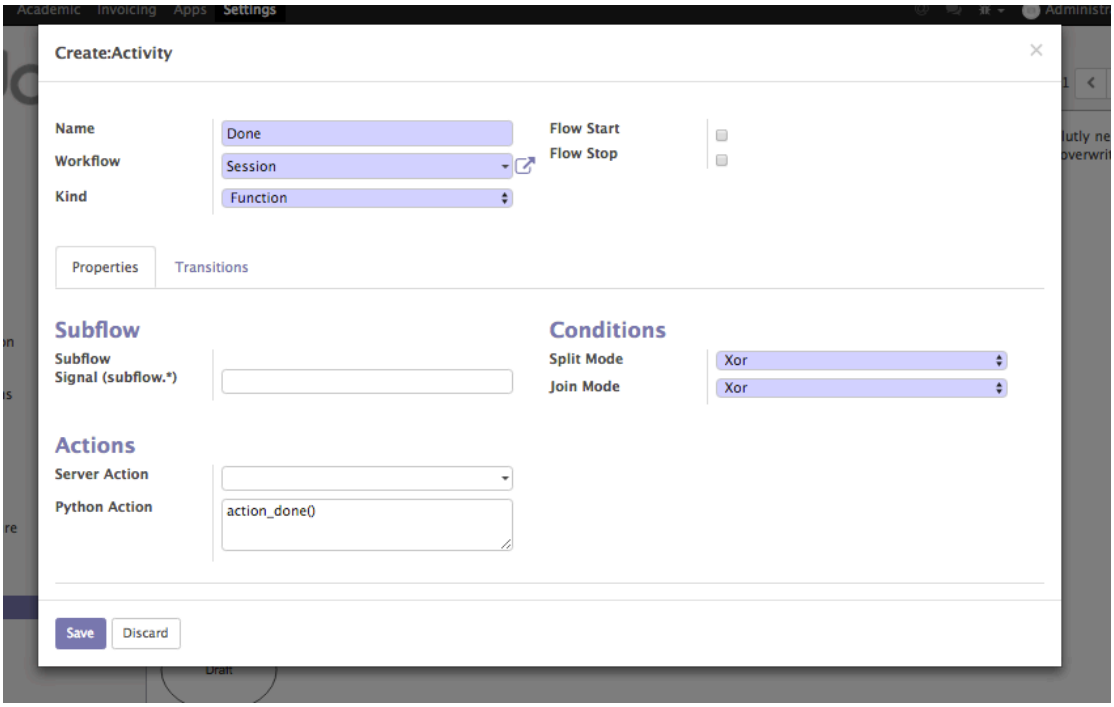
Klik **New Node** lagi:

Name = Done

Kind = Function

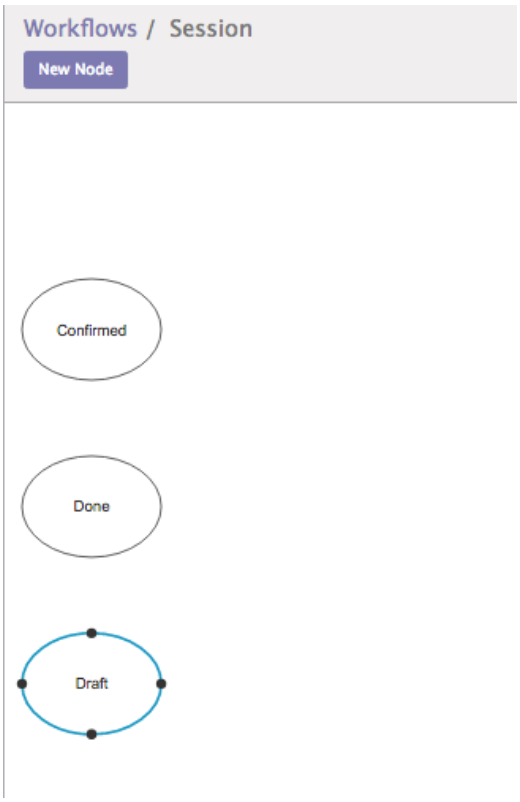
Python Action = action_done ()

Klik **Save**.



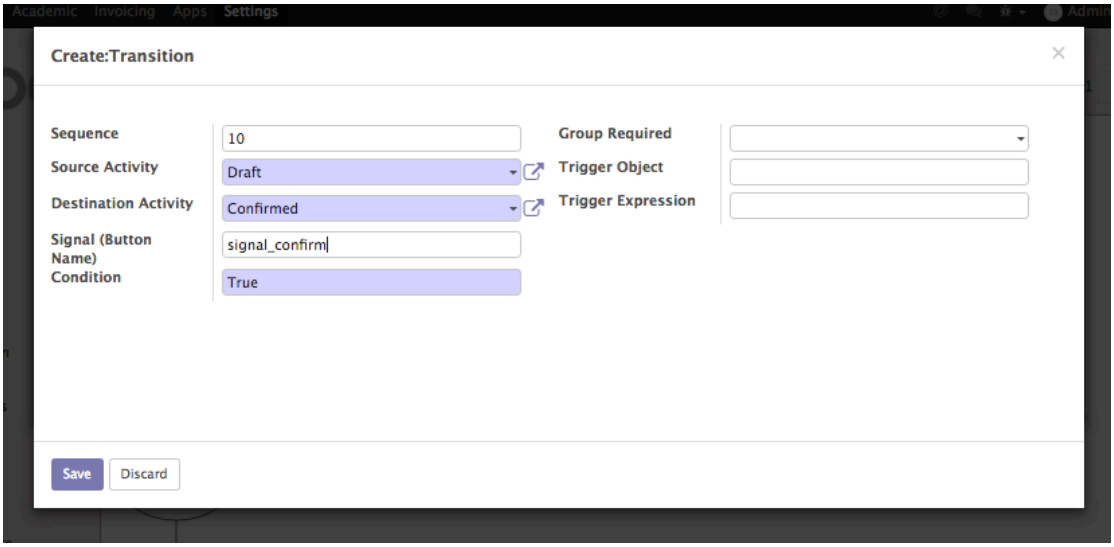
Gambar 41 Tambah node activity Done

Lalu tambahkan panah signal antara node. Caranya klik node. Muncul titik hitam di sekitar node. Tarik panah kearah node yang mau dituju.



Panah dari **Draft → Confirmed**.

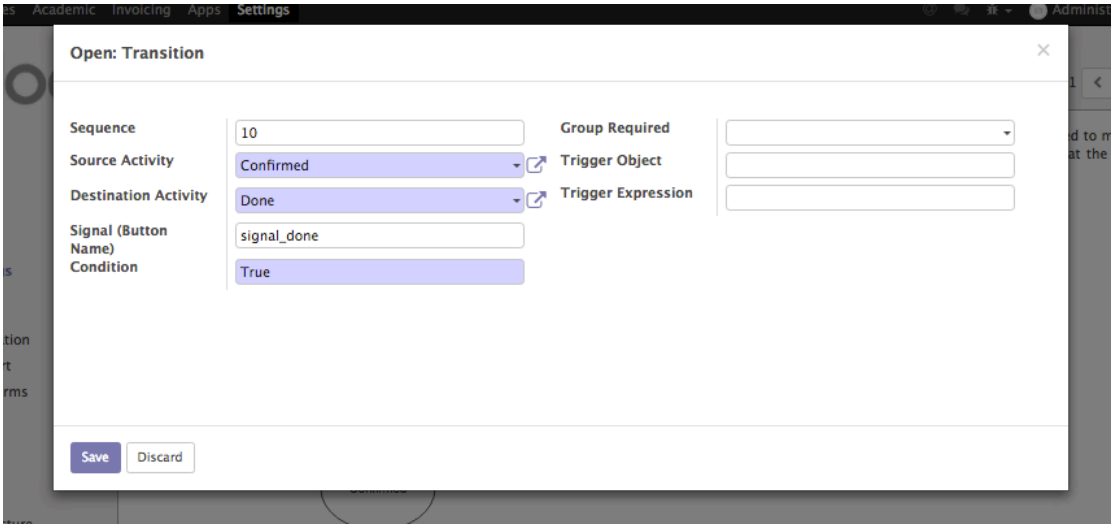
Signal Name = signal_confirm



Gambar 42 Tambah signal confirm

Panah dari **Confirmed → Done**.

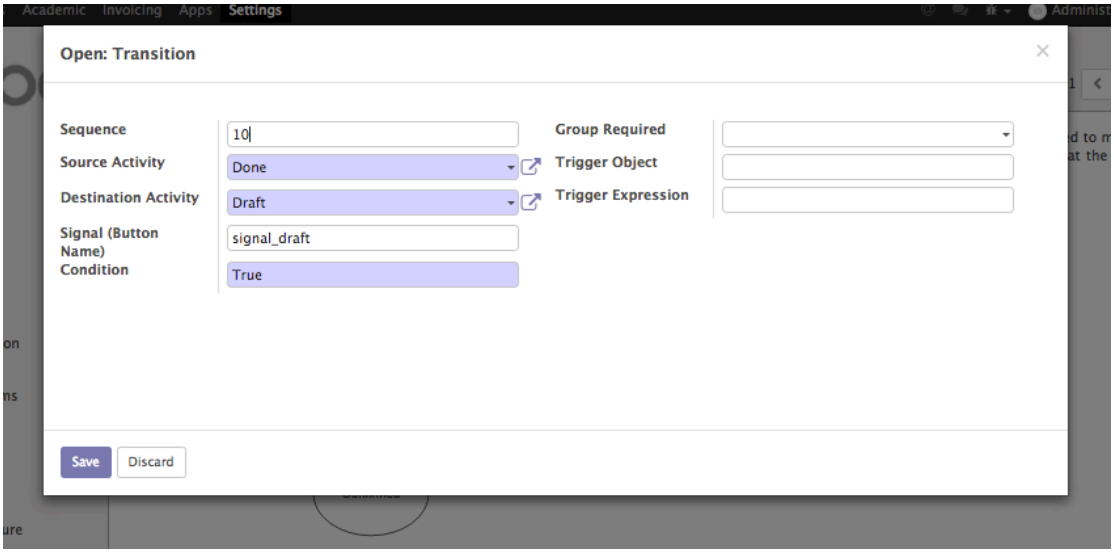
Signal Name = signal_done



Gambar 43 Tambah signal done

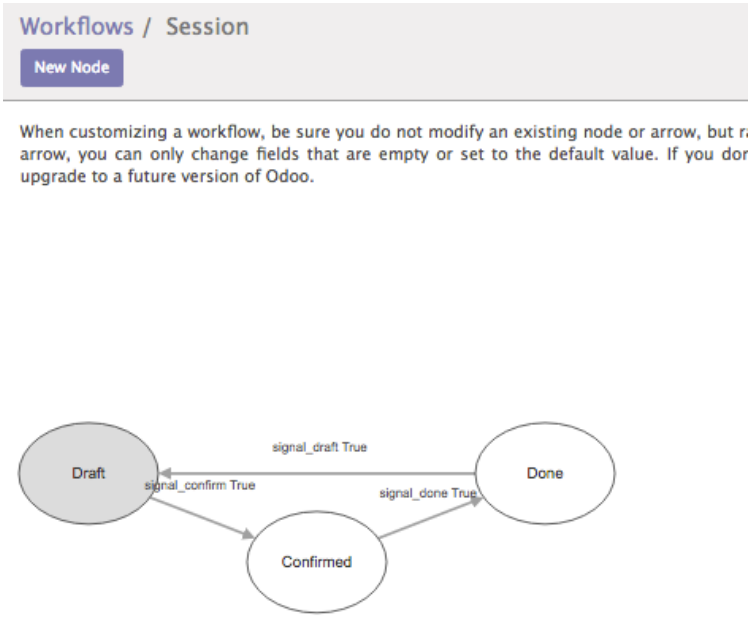
Panah dari **Done → Draft**.

Signal Name = signal_draft



Gambar 44 Tambah signal draft

Setelah semua node dan panah dibuat, hasilnya harus seperti ini.



Gambar 45 Workflow diagram view udah jadi

Lanjut, modif tombol-tombol yang udah dibuat sebelumnya waktu workflow masih statis.

Ganti atribut `name` tombol dengan nama `signal` workflow.

Ganti atribut `type` tombol menjadi `workflow` bukan `object` lagi.

```
<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">
            <header>
                <button string="Confirm" type="workflow"
                    name="signal_confirm"
                    states="draft" />
                <button string="Mark as Done" type="workflow"
                    name="signal_done"
                    states="confirmed" />
                <button string="Reset to draft" type="workflow"
                    name="signal_draft"
                    states="confirmed,done" />

                 <field name="state" widget="statusbar" />
            </header>
```

Gambar 46 Modif button di form view Session

Workflow dinamis hanya berlaku untuk Session record yang baru.

Restart odoo dan update module.

Sekarang workflow udah nggak statis, tapi udah mengikuti alur yang ada pada diagram workflow.

2.3 Export Workflow ke File XML

Supaya nggak ketik ulang XML untuk workflow, kita bisa install module `base_module_record` untuk export workflow yang udah kita buat workflow editor menjadi file XML, supaya bisa kita ikutkan didalam module.

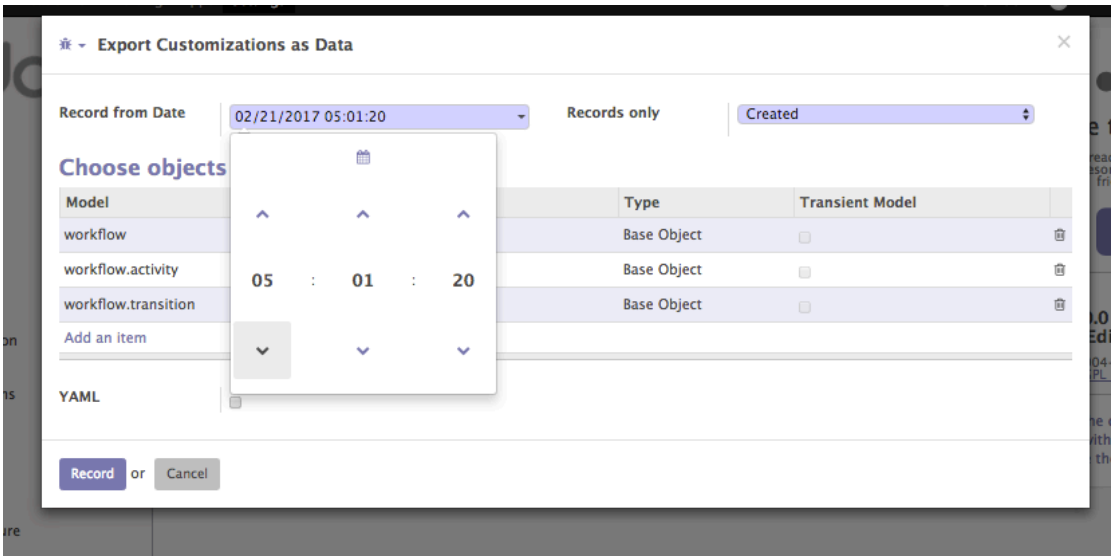
Addons bisa didownload dari sini https://www.odoo.com/apps/modules/10.0/base_module_record/

Selelah berhasil install module `base_module_record`, klik menu `Administration > Customization > Module Creation > Export Customization as Data`

Tentukan `From Date` jadi jam seat kita mulai bikin diagram workflow.

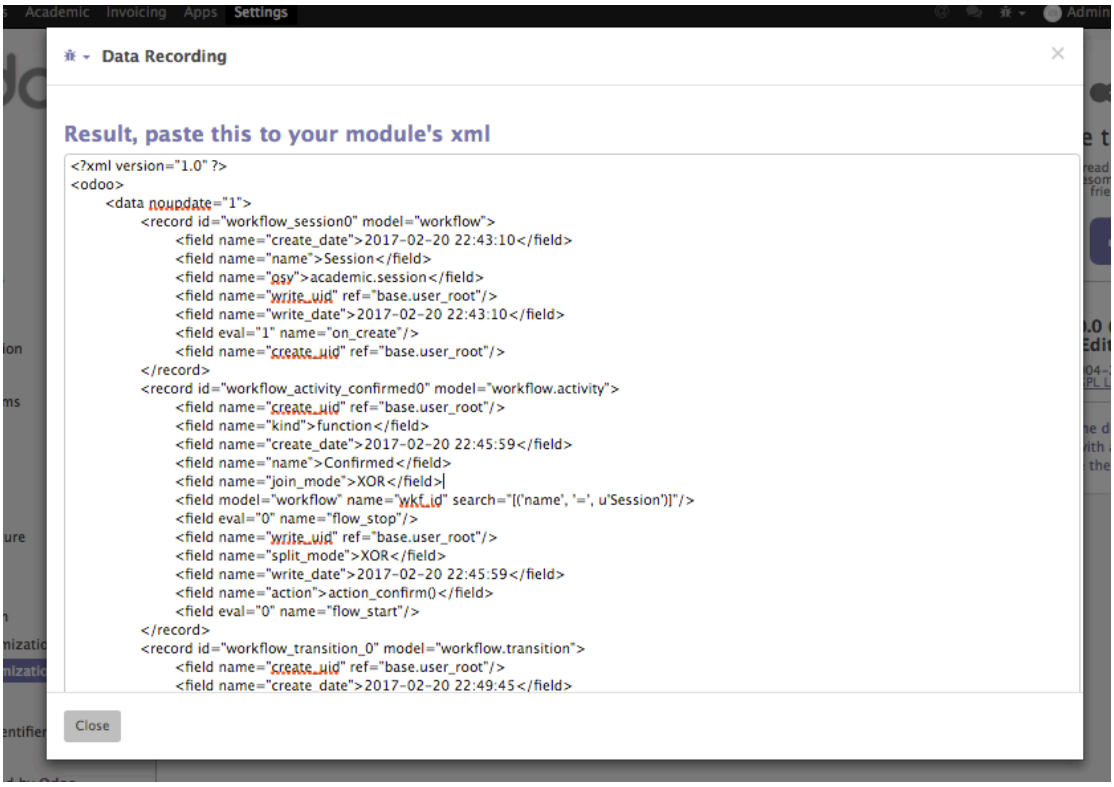
Pilih object `workflow, workflow activity` dan `workflow transition.`

Klik `Record`.



Gambar 47 Export workflow ke XML

Akan muncul dialog pop up box berisi XML yang kita mau.



Gambar 48 Hasil export XML

Copy/paste data XML tersebut ke dalam file baru yang namanya `workflow.xml` dibawah directory module.

Contoh hasil export XML workflow...

```
<?xml version="1.0" ?>
<openerp>
   <data>
      <record id="workflow_session0" model="workflow">
         <field eval="1" name="on_create"/>
         <field name="name">Session</field>
         <field name="osv">academic.session</field>
      </record>

      <record id="workflow_activity_draft0" model="workflow.activity">
         <field name="kind">function</field>
         <field name="name">Draft</field>
         <field name="join_mode">XOR</field>
         <field model="workflow" name="wkf_id"
                search="[('name', '=', u'Session')]"/>
         <field eval="0" name="flow_stop"/>
         <field name="split_mode">XOR</field>
         <field name="action">action_draft()</field>
         <field eval="1" name="flow_start"/>
      </record>

      <record id="workflow_activity_confirmed0" model="workflow.activity">
         <field name="kind">function</field>
         <field name="name">Confirmed</field>
         <field name="join_mode">XOR</field>
         <field model="workflow" name="wkf_id"
                search="[('name', '=', u'Session')]"/>
         <field eval="0" name="flow_stop"/>
         <field name="split_mode">XOR</field>
         <field name="action">action_confirm()</field>
         <field eval="0" name="flow_start"/>
      </record>

      <record id="workflow_activity_done0" model="workflow.activity">
         <field name="kind">function</field>
         <field name="name">Done</field>
         <field name="join_mode">XOR</field>
         <field model="workflow" name="wkf_id"
                search="[('name', '=', u'Session')]"/>
         <field eval="0" name="flow_stop"/>
         <field name="split_mode">XOR</field>
         <field name="action">action_done()</field>
         <field eval="0" name="flow_start"/>
      </record>

      <record id="workflow_transition_0" model="workflow.transition">
         <field name="signal">signal_done</field>
         <field model="workflow.activity" name="act_from"
                search="[('name', '=', u'Confirmed')]"/>
         <field model="workflow.activity" name="act_to"
                search="[('name', '=', u'Done')]"/>
         <field name="condition">True</field>
      </record>

      <record id="workflow_transition_1" model="workflow.transition">
         <field name="signal">signal_confirm</field>
         <field model="workflow.activity" name="act_from"
```

```
                search="[('name', '=', u'Draft')]"/>
            <field model="workflow.activity" name="act_to"
                search="[('name', '=', u'Confirmed')]"/>
            <field name="condition">True</field>
        </record>

        <record id="workflow_transition_2" model="workflow.transition">
            <field name="signal">signal_draft</field>
            <field model="workflow.activity" name="act_from"
                search="[('name', '=', u'Done')]"/>
            <field model="workflow.activity" name="act_to"
                search="[('name', '=', u'Draft')]"/>
            <field name="condition">True</field>
        </record>

    </data>
</openerp>
```

Struktur file di folder addons sejauh ini...

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

Cantumkan file XML ini pada file `__openerp__.py`.

```
{
    "name": "Academic Information System Day 3",
    "version": "1.0",
    "depends": [
        "base",
        "account",
        "sale",
    ],
    "author": "akhmad.daniel@gmail.com",
    "category": "Education",
    'website': 'http://www.vitraining.com',
    "description": """\
Academic Information System Day 3

""",
    "data": [
        "menu.xml",
        "course.xml",
        "session.xml",
```

```
        "attendee.xml",
        "partner.xml",
        "workflow.xml",
    ],
    "installable": True,
    "auto_install": False,
     "application": True,
}
```

Gambar 49 Panggil workflow.xml dari __openerp__.py

3 REKAP HARI 3

Gimana gan,… masih semangat? ☺

Berikut rekap materi untuk hari ke -3 …

Advanced View

Warna List View, Calendar View, Search View, Gantt View, Chart/ Graph View, Kanban, Field Image di Session, Related Field

Workflow

Workflow Statis , Workflow dinamis, Workflow export ke XML