

5 HARI MENGUAK MISTERI ADDONS ODOO 10

AKHMAD DANIEL SEMBIRING
2017 - VITRAINING.COM

REVISI FEB. 19, 17

5 HARI MENGUAK MISTERI

ADDONS ODOO 10

1	HARI 1: Pembukaan	6
1.1	Target pencapaian 5 Hari.....	6
1.2	Contoh Soal Aplikasi : Academic Information System.....	7
1.3	Tampak Jadi	7
2	Desain ERD	18
2.1	Course	18
2.2	Session.....	18
2.3	Attendee.....	19
2.4	Instructor Partner	19
3	Persiapan	20
3.1	Install Database	20
3.2	Activate Developer Mode	20
3.3	Start Server dan Update Module.....	21
4	Struktur Addons	23
4.1	Struktur file	23
4.1.1	File <code>__openerp__.py</code>	23
4.1.2	File <code>__init__.py</code>	23
4.2	Penempatan Folder.....	24
5	Bikin Addons Academic	25
5.1	Bikin folder	25
5.2	Bikin file <code>__openerp__.py</code>	25
5.3	Bikin file <code>__init__.py</code>	25
5.4	Reload odoo.....	26
5.4.1	Linux	27
5.4.2	Windows	27
6	Bikin Menu dan ActionWindow	28
6.1	Bikin Menu.....	28
6.2	Bikin Action Window	30
6.3	Update <code>__openerp__.py</code>	32
7	Class Course	34
7.1	Bikin Class Course	34
7.2	Bikin Tree View Course.....	39
7.3	Bikin Form View Course	41
8	Class Session	46
8.1	Menu dan Action Window Session.....	46
8.2	Bikin Class Session	47
8.3	Modif List View Session	51
8.4	Modif Form View.....	54
8.5	Bikin Relasi Course Has Many Session	56
8.6	Modif View Form Course – Tambah Session.....	58
9	Class Attendee	61
9.1	Menu dan Action Window Attendee.....	61

9.2	Bikin Class Attendee	63
9.3	Modif List View Attendee	66
9.4	Bikin Relasi Session Has Many Attendee.....	69
9.5	Modif Form View Session – Tambah Attendee	71
10	Rekap Hari 1	74
11	HARI 2: Inheritance – Instructor	75
11.1	Inherit Partner Class.....	75
11.2	Inherit Partner View.....	76
11.3	Modif Form Session – Filter Partner is instructor	81
11.4	Modif Form Session – Filter Partner Category	83
11.5	Menu Instruktur	87
12	Functional Fields – Percentage Taken Seats	90
12.1	Definisi function fields	90
12.2	Menampilkan di tree dan view.....	92
12.3	ProgressBar di form.....	94
12.4	Progress bar di tree view.....	95
13	Event OnChange	97
14	Constraints	98
14.1	Python Constraints.....	98
14.2	SQL contrains.....	100
15	Nilai Default – Lambda Function	104
15.1	Apa Itu Lambda Function?	106
15.2	Parameter function *x ?	106
16	Fitur Duplicate	108
17	Rekap Hari 2	111
18	HARI 3: Advanced View	112
18.1	Warna List View	112
18.2	Calendar View.....	113
18.3	Search View	115
18.4	Gantt View	119
18.5	Chart/ Graph View.....	121
18.6	Kanban	123
18.7	Nambahin Field Image di Session	127
18.8	Related Field – Apa Nama Course suatu Attendee ?	130
19	Workflow	133
19.1	Workflow Statis	133
19.2	Dynamic Workflow – Bikin Diagram Workflow	137
19.3	Export Workflow ke File XML.....	145
20	Rekap Hari 3	149
21	HARI 4: Security	150
21.1	Bikin Group Lewat Interface	150
21.2	Bikin Group Lewat XML	153
21.3	Masukkan User ke Group.....	155
21.4	Import CSV Access Right.....	156
21.5	Record Rules	160
21.6	Hubungan Group Workflow	164
22	Wizard	167
22.1	Definisikan Class Wizard	167

22.2	Bikin Menu Untuk wizard.....	169
22.3	Modif form view Wizard	172
22.4	Bikin Method untuk Memproses Data Wizard	173
22.5	Bikin Menu Context.....	177
	22.5.1 Edit file <code>wizard/create_attendee_view.xml</code>	177
	22.5.2 Edit file <code>wizard/create_attendee.py</code>	178
22.6	Add an onchange method.....	180
22.7	Wizard untuk Banyak Session Sekaligus	180
	22.7.1 Edit file <code>wizard/create_attendee.py</code>	180
	22.7.2 Edit file XML <code>wizard/create_attendee.xml</code>	181
23	Rekap Hari 4	183
24	HARI 5: Internationalization	184
24.1	Bikin Direktori i18n	184
24.2	Bahasa Udah ada di odoo.....	184
	24.2.1 Install Bahasa Target.....	184
24.3	Bahasa Belum Ada di odoo	185
	24.3.1 Bikin Template Translate academic.pot	185
	24.3.2 Import ke odoo	186
24.4	Sinkronisasi Istilah.....	187
24.5	Bikin File Template per Bahasa.....	188
24.6	Terjemahin File Template Bahasa	189
24.7	Reload Bahasa Indonesia.....	190
24.8	Istilah Tambahan	191
25	Report RML	193
25.1	Instalasi	193
25.2	Install Plug-in di OpenOffice.....	193
25.3	Konfigurasi.....	195
25.4	Bikin Report Baru	196
25.5	Syntax Template	199
25.6	Modify Existing Report.....	200
25.7	Report di Addons – RML	201
26	Report Webkit	203
26.1	Instalasi	203
26.2	Test Bikin Report Manual.....	204
	26.2.1 Create Template	204
	26.2.2 Bikin Action Button.....	205
26.3	Install Report dari Addons.....	206
	26.3.1 Buat File Template	206
	26.3.2 Buat XML	207
27	Report QWEB	209
27.1	Buat folder report	209
27.2	Buat File XML report untuk Menu Report.....	209
27.3	XML record untuk Template QWEB.....	210
27.4	XML record untuk report per record session	210
28	Dashboard	216
28.1	Bikin Sub Menu Dashboard	216
28.2	Tambahi dashboard XML.....	217
29	Web Services	222
29.1	Instalasi XML-RPC for PHP	222
29.2	Aktifkan PHP Curl Module	222

29.3	Setup folder aplikasi	222
29.4	Login	224
29.5	Search.....	225
29.6	Read	227
29.7	Create	230
29.8	Delete	232
29.9	Write.....	234
29.10	Write one2many Fields	237
30	Rekap Hari 5	254
31	Penutupan	255
32	Referensi	256
33	Tentang Penulis	257

1 HARI 1: PEMBUKAAN

Buku ini menceritakan bagaimana mulai membuat addons untuk odoo 10 secara lengkap. Pembahasan dimulai dari membuat folder, membuat file identifikasi addon, membuat class, membuat view XML, inherit class, inherit view, advanced view, workflow, security, wizard, dashboard dan report, bahasa, sampai dengan web services.

1.1 TARGET PENCAPAIAN 5 HARI

HARI 1:

- Intro
- Contoh Soal Aplikasi : Academic Information System
- Struktur Addons
- Desain ERD
- Bikin Addons Academic
- Bikin Menu dan ActionWindow
- Class Course
- Class Session
- Relasi Course ke Session
- Class Attendee
- Relasi Session ke Attendee

HARI 2:

- Inheritance – Instructor
- Functional Fields – Percentage Taken Seats
- Event OnChange
- Constraints
- Nilai Default – Lambda Function
- Fitur Duplicate

HARI 3:

- Advanced View
- Workflow

HARI 4:

- Security
- Wizard

HARI 5:

- Internationalization
- Report
- Dashboard
- Web Services

1.2 CONTOH SOAL APLIKASI : ACADEMIC INFORMATION SYSTEM

Contoh soal yang kita jadikan bahan praktek adalah system informasi Akademik. Terdiri dari data **Course** yang punya banyak **Session**. Setiap **Session** dihadiri oleh banyak peserta (**Attendee**).

Course ada penganggungjawabnya, yang kita link ke **User** odoo. Setiap **Session** ada instrukturnya yang link ke **Partner** odoo.

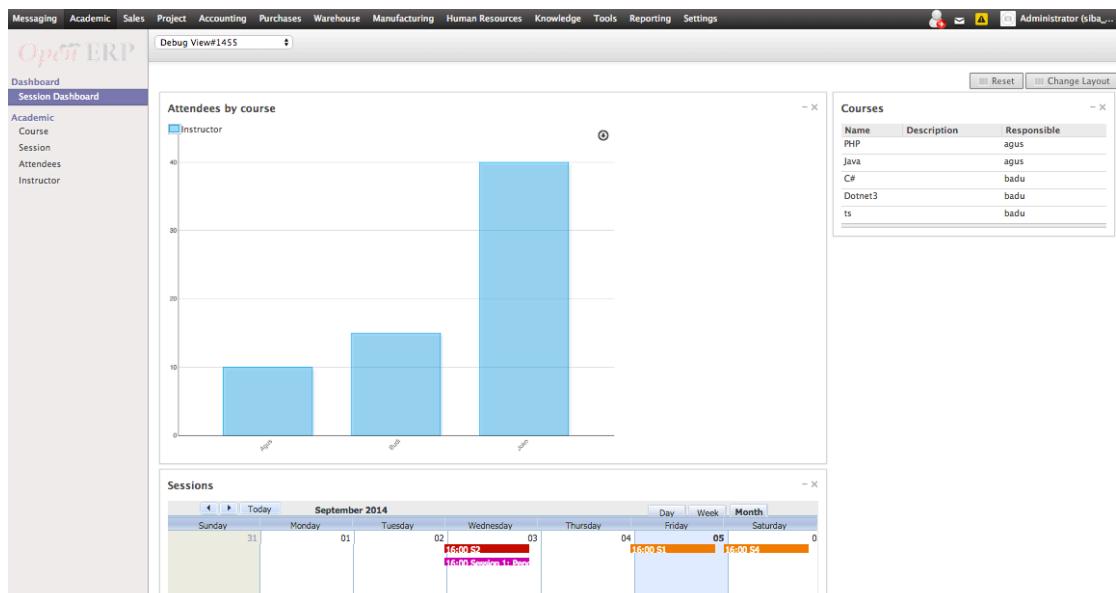
Setiap **Attendee** dihadiri oleh peserta yang juga di-link **Partner** odoo.

Partner yang udah jadi **Instruktur** pada suatu **Session** nggak boleh lagi jadi **Attendee** pada **Session** itu.

1.3 TAMPAK JADI

Berikut ini tampilan tampak jadi aplikasi Academic information system yang akan kita bangun sama-sama..

Menu utama dan tampilan awal dashboard



Gambar 1 Tampilan dashboard

Menu Daftar Course yang menampilkan daftar Course yang tersedia.

The screenshot shows the 'Daftar Course' (List Courses) page. The top navigation bar includes links for Messaging, Academic, Sales, Project, Accounting, Purchases, Warehouse, Manufacturing, and More. The 'Academic' section on the left is selected and shows sub-links: Course, Session, Attendees, Add attendee, and Instructor. The main content area has a search bar, a 'Create' button, and a table with columns for Name and Description. There are four entries in the table: PHP, Java, Dotnet, and ts.

Name	Description
PHP	
Java	
Dotnet	
ts	

Gambar 2 Menu daftar Course

Aplikasi ini mendukung Bahasa Indonesia.. dan bahasa apapun yang kita mau translate.

Gambar 3 Tampilan dalam Bahasa Indonesia

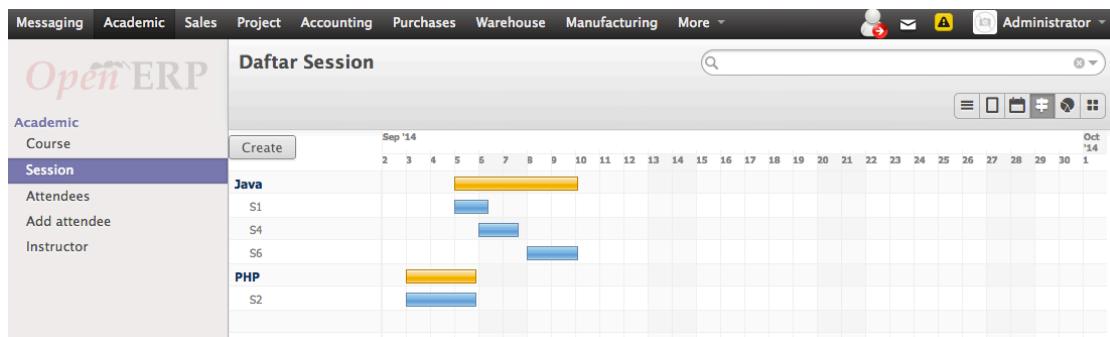
Daftar Session, tampil dalam warna sesuai kondisi tertentu.
Session bisa di-grouping per Course dan Tanggal. Bisa juga di-filtering sesuai kondisi tertentu.

Gambar 4 Tampilan daftar Session

Contohnya grouping Session per Course dan Start Date...

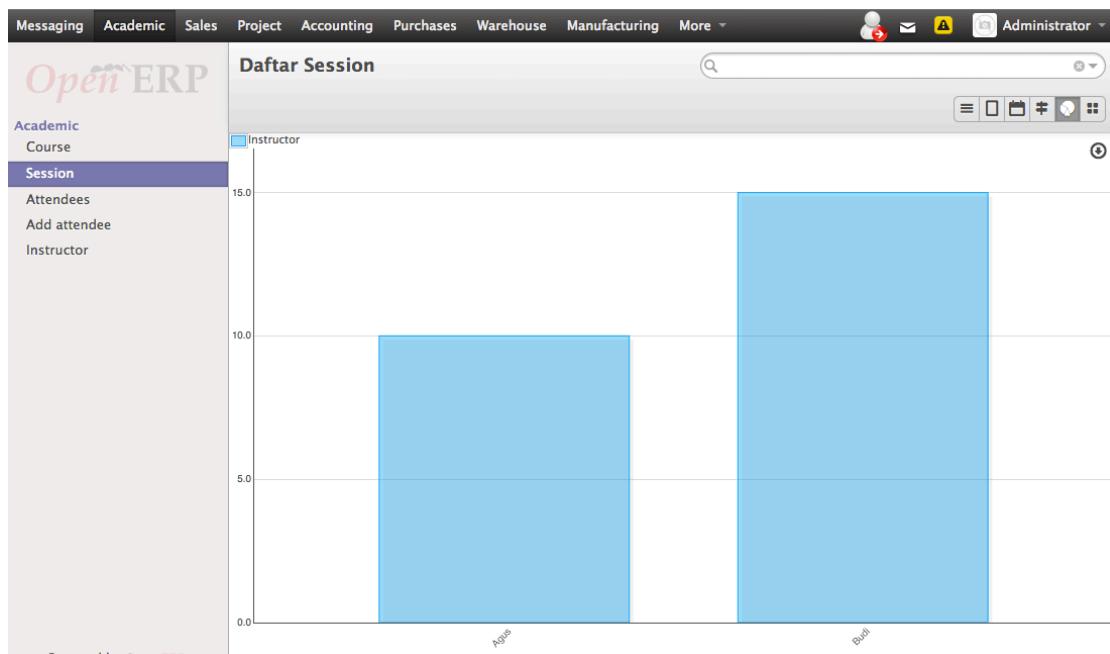
Gambar 5 Grouping Session per Course dan Start Date

Session bisa ditampilkan dalam Gantt Chart...



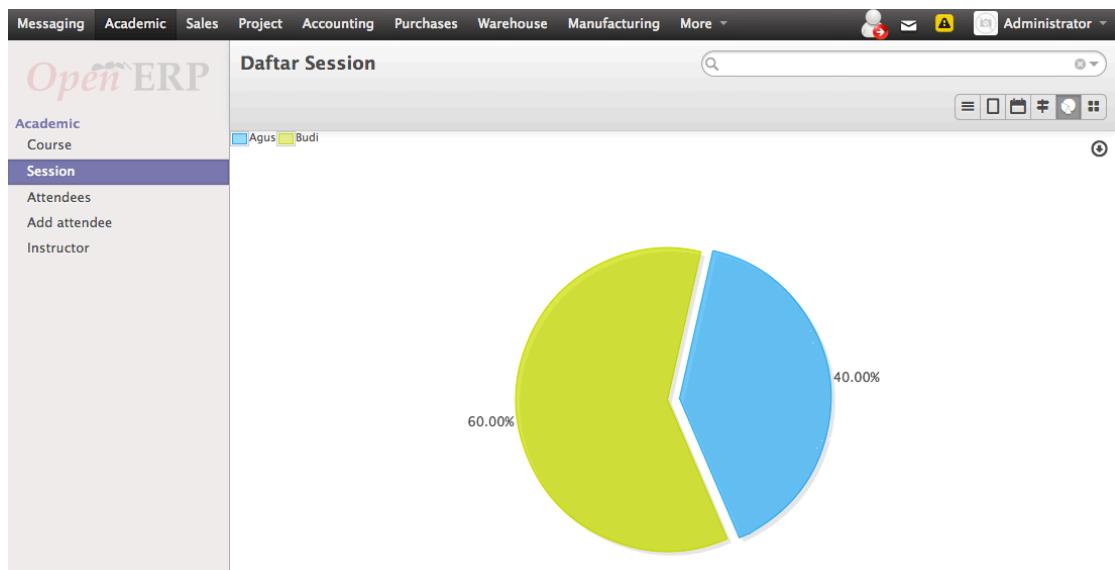
Gambar 6 Tampilan gantt chart

Session bisa disajikan dalam Graph, contohnya grafik total Durasi per Instuktur Session...



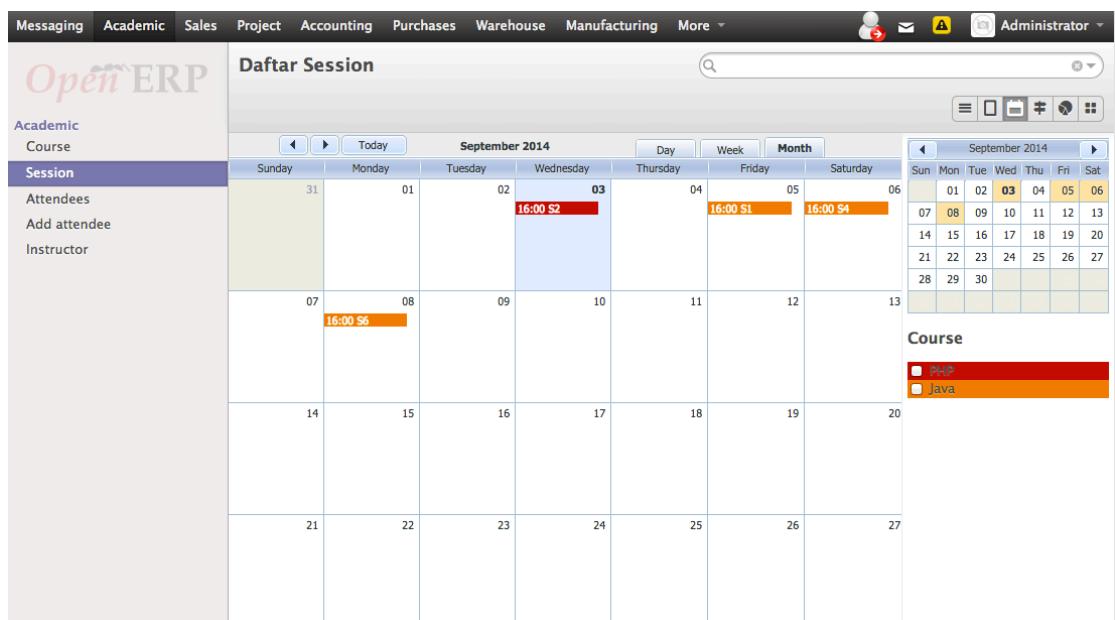
Gambar 7 Grafik Total Durasi per Instruktur Session

Grafik bisa tampil dalam Pie Chart...



Gambar 8 Total durasi per instruktur dalam Pie hart

Session bisa ditampilkan dalam Calendar...



Gambar 9 Calendar view session

Session bisa ditampilkan dalam Kanban, dikelompokkan dalam Course...

Session	Type	Seats	Taken Seats
S2	PHP	5	60.00
S1	Java	3	40.00
S4	Java	10	0.00
S6	Java	5	0.00

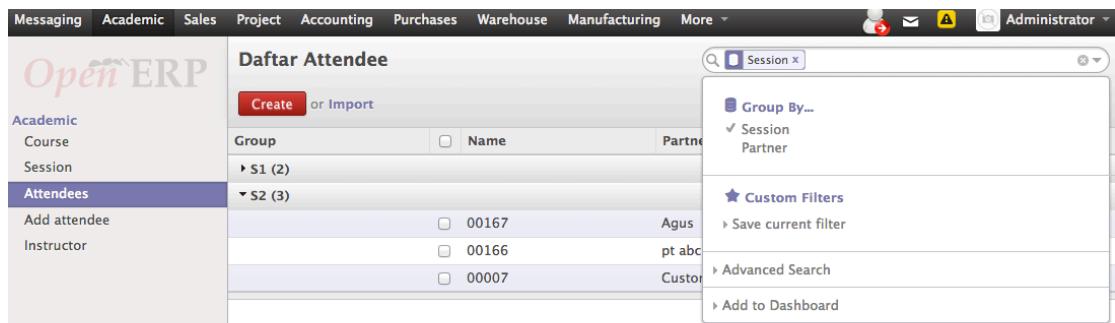
Gambar 10 Session Kanban View

Menu Daftar Instruktur, yang merupakan inherit dari Partner dengan penambahan field `is_instructor`...

Instructor	Sales
Agus	1 Sales
Budi	2 Sales

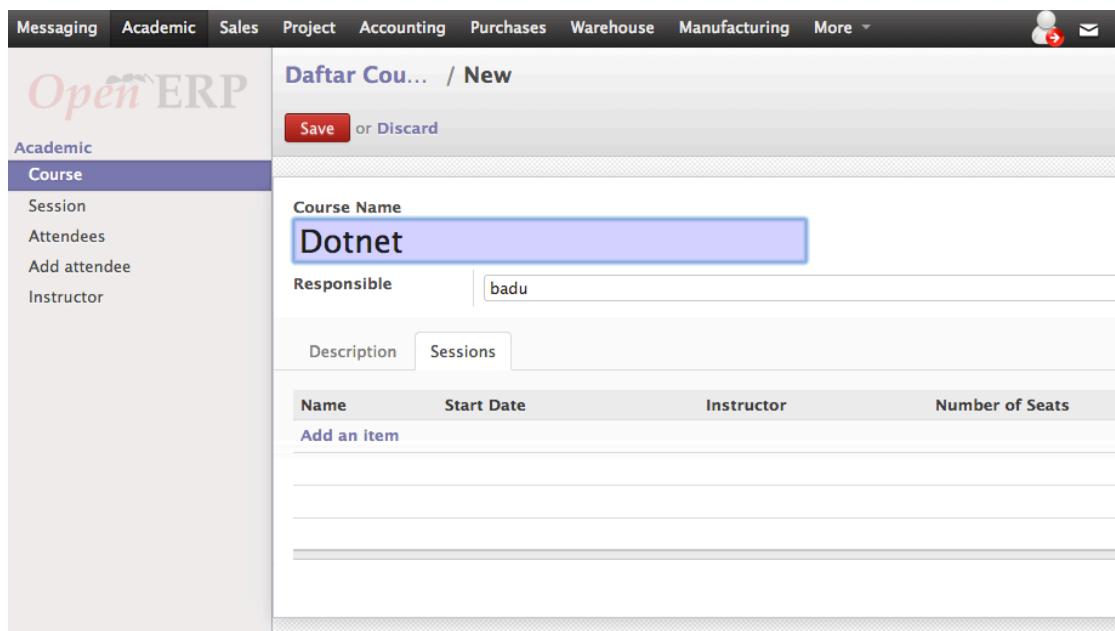
Gambar 11 Daftar instruktur

Menu Daftar Attendee, bisa dikelompokkan dalam Group atau Partner...



Gambar 12 Daftar Attendee

Membuat Course, bisa langsung bikin Session disitu...



Gambar 13 Create Course

Create Session, terdapat Constraint jika Attendee dan Instruktur sama partnernya, ada warning message jika Number of Seats nilai negative atau kurang dari jumlah Attendee yang udah ada saat itu...

Create: Sessions

Confirm Draft Confirmed Done

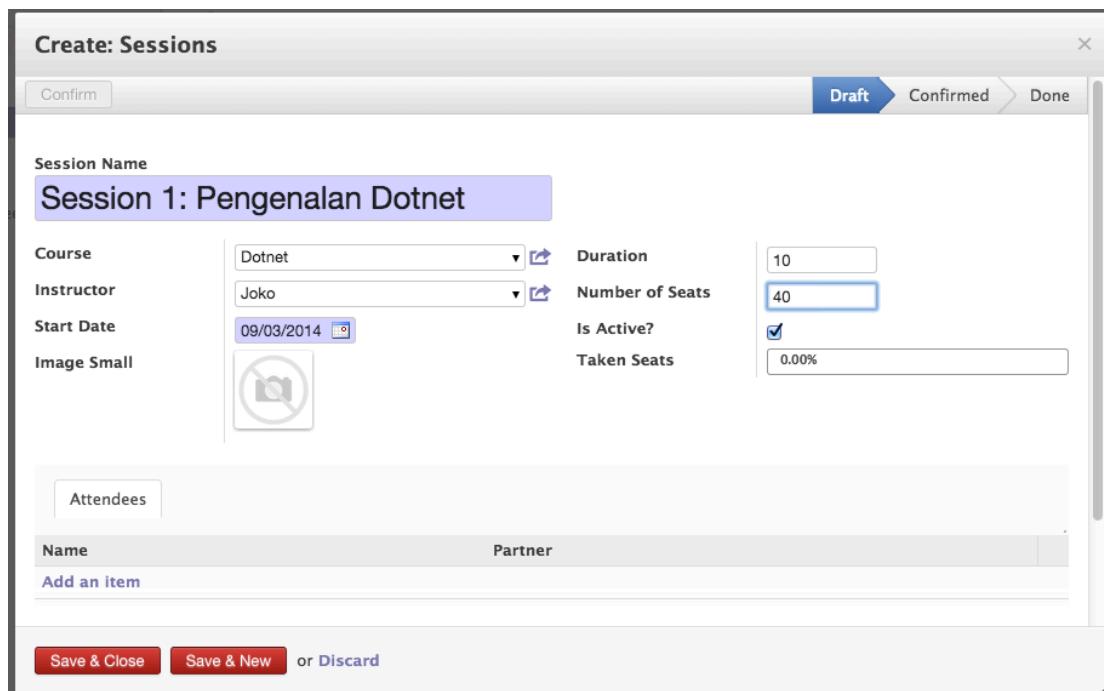
Session Name
Session 1: Pengenalan Dotnet

Course: Dotnet Duration: 10
Instructor: Joko Number of Seats: 40
Start Date: 09/03/2014 Is Active?:
Image Small: 

Attendees

Name	Partner
Add an item	

Save & Close Save & New or Discard



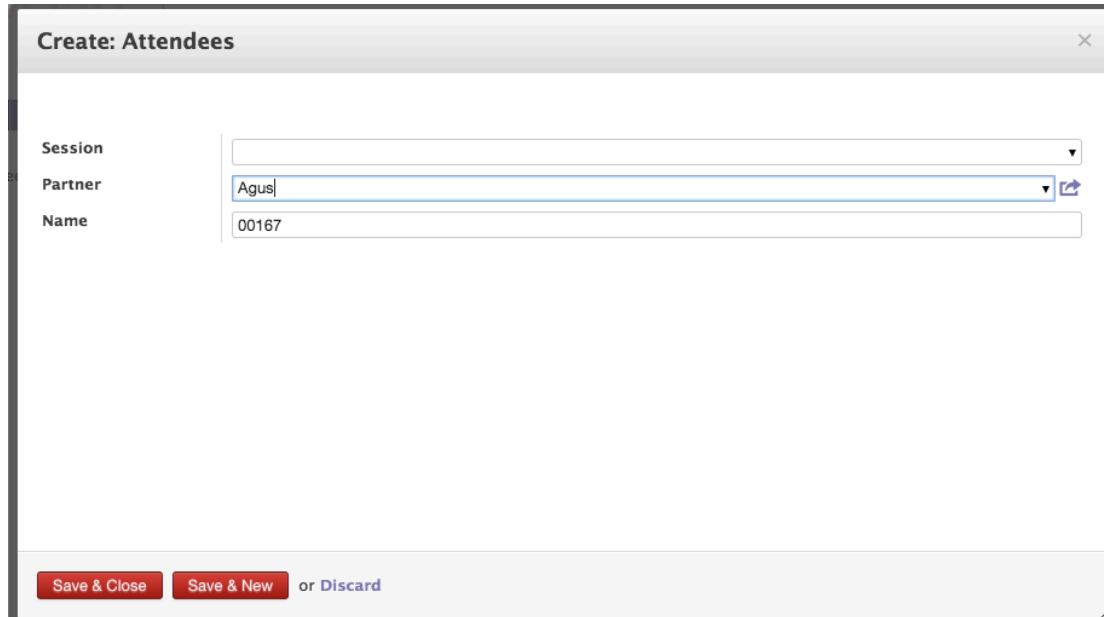
Gambar 14 Create Session

Create Attendee; ada event onchange di Partner yang otomatis mengisi kolom Name dengan ID partner diformat 5 digit.

Create: Attendees

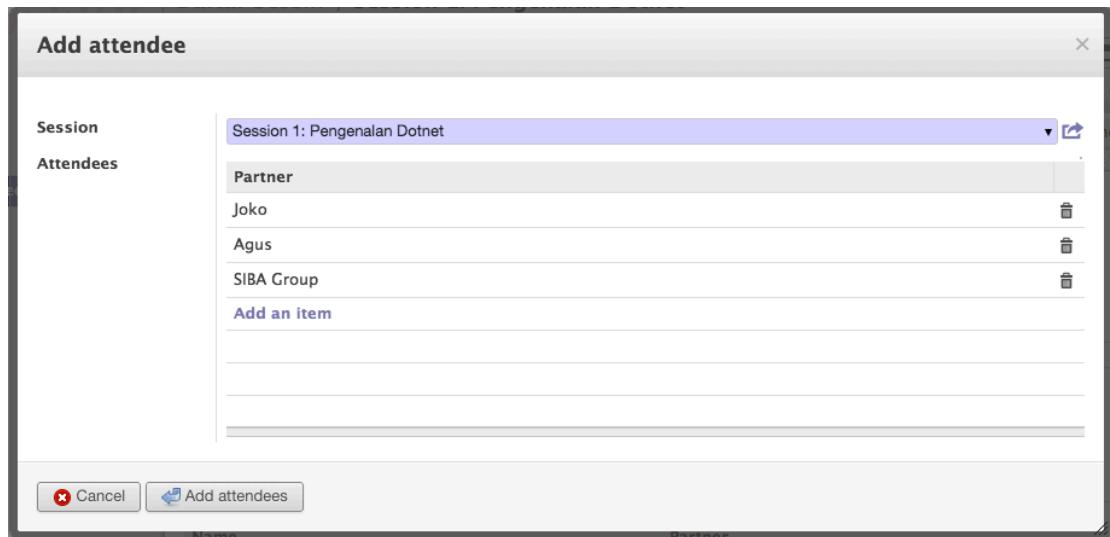
Session:
Partner: Agus
Name: 00167

Save & Close Save & New or Discard



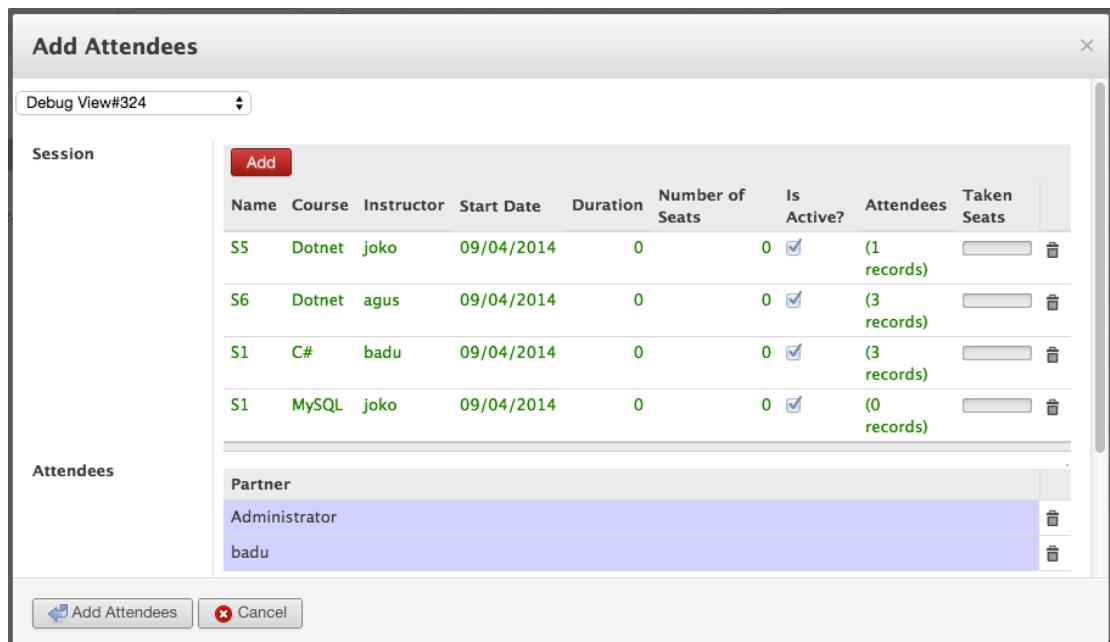
Gambar 15 Create Attendee

Create Attendee menggunakan Wizard supaya bisa create Attendee sekaligus banyak, nggak satu per satu ...



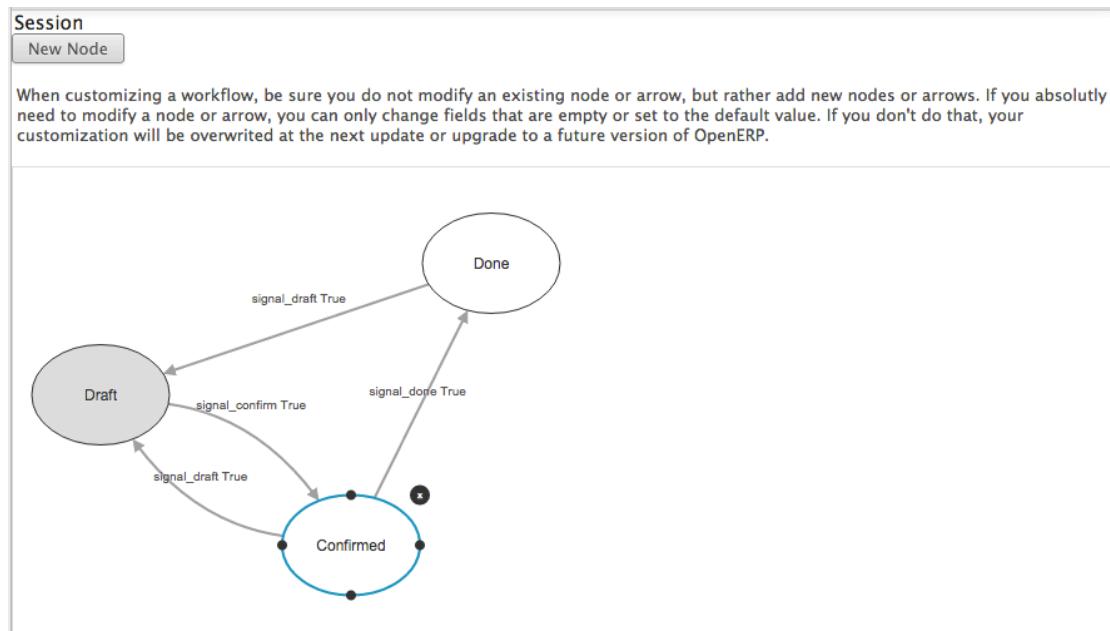
Gambar 16 Add Attendee wizard

Atau bisa juga daftarin banyak Attendee sekaligu ke banyak Session...



Gambar 17 Add Attendees ke banyak Session

Terdapat Workflow dinamis untuk mengubah status Session mulai dari Draft, Confirmed, dan Done...



Gambar 18 Workflow diagram

Ada group access security, yaitu group Academic/Manager dan Academic/User..

Groups / Academic / Manager

Application		Name	Academic / Manager		
Share Group	Portal	<input type="checkbox"/>	<input type="checkbox"/>		
Users	Inherited	Access Rights	Rules	Notes	
Object	Read Access	Write Access	Create Access	Delete Access	Name
academic.course	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	course_manager
academic.session	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	session_manager
academic.attendee	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	attendee_manager

Gambar 19 Group access rights

Report untuk menghasilkan laporan dalam PDF ...

Phone: 12345678, 89900020
Mail: info@yourcompany.com

Session Report

Java - S1

From 09/05/2014.

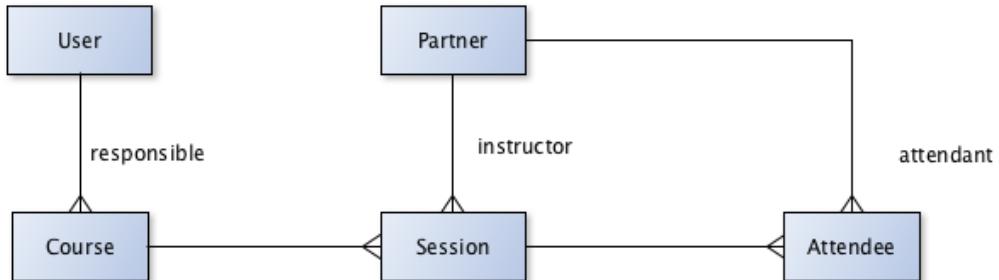
Attendees:

- Agus
- Joko
- Joko
- Agus
- Cabang A
- Customer
- Administrator
- Administrator
- Joko

Gambar 20 Laporan PDF

2 DESAIN ERD

Berikut gambar entity relationships diagram sesuai spesifikasi yang udah dibahas diatas...



Gambar 21 ERD sistem Academic

2.1 COURSE

Berikut ini detail field table Course. Setiap Course ada penanggungjawabnya (responsible) yang link ke table User odoo.

Field	Type	Keterangan
name	Char, 100	Nama Course
description	Text	Keterangan lengkap
responsible_id	FK many2one ke table user	Siapa penanggung jawab Course ini
session_ids	One2many ke table Session	Satu Course bisa punya banyak Session

2.2 SESSION

Berikut ini detail field table Session, table yang mencatat Session yang dimiliki oleh suatu Course.

Setiap Session ada Instructor-nya yang link ke table Partner odoo.

Field	Type	Keterangan
name	Char, 100	Nama Course
course_id	Text	Keterangan lengkap
instructor_id	FK many2one ke table Partner	Siapa Instruktur Session ini
start_date	Date	Tanggal mulai Session
duration	Integer	Berapa hari durasi Session
seats	Integer	Berapa maksimal kapasitas peserta
active	Boolean	Apakah aktif atau nggak
attendee_ids	One2many ke table Attendee	Satu Session bisa punya banyak Attendee
taken_seats	Function Field	Dihitung berdasarkan seats dan jumlah peserta setiap seat
state	Enum (Draft, Confirmed, Done)	Status session terkait workflow
image_small	Binary	Bisa upload image langsung

2.3 ATTENDEE

Berikut ini detail field table Attendee, table yang mencatat siapa-siapa aja Partner OperERP yang menghadiri suatu Session.

Field	Type	Keterangan
name	Char, 100	Nama Attendee, boleh diisi dengan nomor pendaftaran
session_id	FK many2one ke table Session	Attendee untuk Session yang mana
partner_id	FK many2one ke table Partner	Siapa partner yang jadi Attendee

2.4 INSTRUCTOR PARTNER

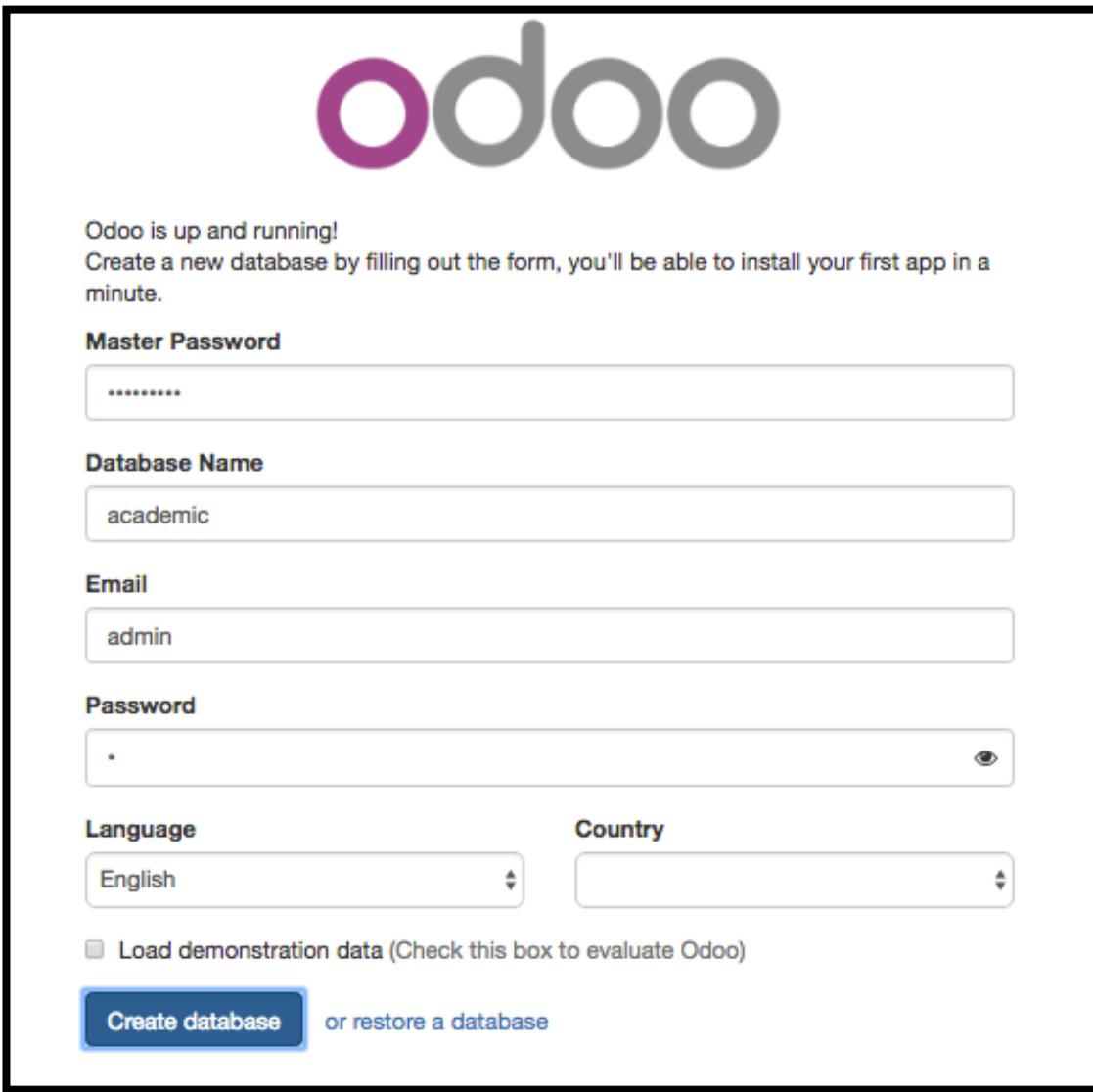
Instructor diturunkan dari table Partner, jadi memiliki semua field yang ada di Partner, dengan tambahan field berikut ini...

Field	Type	Keterangan
is_instructor	Boolean	Apakah partner ini instruktur atau bukan

3 PERSIAPAN

3.1 INSTALL DATABASE

Install database odoo baru. Dari halaman login odoo klik link [Manage Databases](#). Jika belum ada database, maka langsung masuk ke halaman Manage Database.

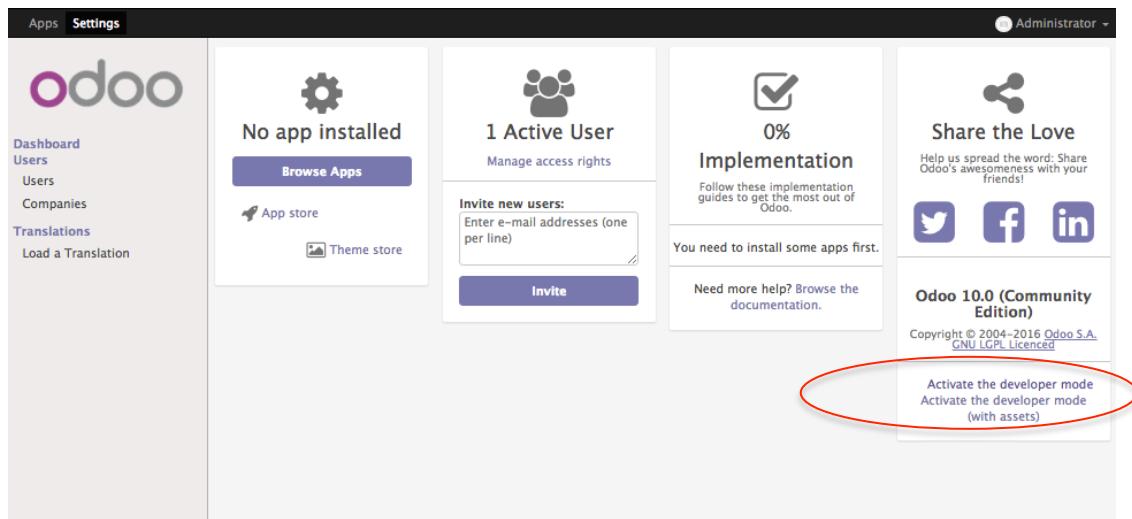


Gambar 22 Bikin database baru

Buat database dengan nama yang dikehendaki misalnya **academic**.

3.2 ACTIVATE DEVELOPER MODE

Login ke database yang baru dibuat, activate Developer Mode melalui menu Settings agar muncul semua fitur teknikal Odoo.



Gambar 23 Activate Developer Mode

3.3 START SERVER DAN UPDATE MODULE

Selama development kita akan menjalankan odoo server secara manual melalui terminal Linux atau command prompt Windows.

Disini kita perlu tentukan file konfigurasi yang akan digunakan melalui argument `-c`, supaya kita bisa atur konfigurasi waktu development lebih flexible dan cepat.

```
./odoo-server -c odoo-server.conf
```

Isi dari file konfigurasi `odoo-server.conf` yang paling penting:

```
db_host = localhost      # ip address server db postgres
db_port = 5432            # port db postgress
db_user = odoo            # username postgres
db_password = odoo        # password user postgres
addons_path = odoo/addons/,/Users/odoo/addons/  #lokasi addons
#logfile=/tmp/openrp.log   # kalo ada logfile, ditutup dulu pake #
```

Parameter yang perlu diperhatikan adalah `logfile` yaitu nentuin kemana informasi log server odoo akan ditampilkan. Kalo isinya berupa file seperti contoh diatas ke file `/tmp/odoo.log`,

maka perlu kita disable dulu selama development biar informasi log nya muncul langsung di terminal.

Cara disablenya seperti di atas, yaitu dikasi tanda # yang artinya dibuat jadi comment line.

4 STRUKTUR ADDONS

4.1 STRUKTUR FILE

Addons odoo tersimpan dalam satu folder yang berada di folder addons server odoo.

Addons terdiri dari komponen-komponen :

Business Object: berupa class Python yang merupakan turunan dari models.Model, class bawaan odoo/OpenObject.

Data: bisa berupa file XML atau CSV yang berisi meta-data (view dan workflow), konfigurasi parameter module, atau data demo.

Wizard: kotak dialog yang berguna untuk memudahkan user dalam menginput data.

Report: bisa berupa QWEB template, yaitu untuk menampilkan report dari semua data untuk ditampilkan dalam HTML atau PDF.

Setiap addons modul harus punya file `__openerp__.py` dan `__init__.py`. kalo nggak, folder tersebut nggak dianggap sebagai modul oleh odoo.

4.1.1 FILE `__OPENERP__.PY`

Adalah tempat untuk kita mendefinisikan segala informasi tentang modul addons, seperti nama, keterangan, daftar modul odoo lain yang harus ada, referensi ke file XML atau CSV yang diperlukan oleh module. Data itu dituliskan dalam satu dictionary Python.

4.1.2 FILE `__INIT__.PY`

Adalah file Python module descriptor berisi semua file Python yang diperlukan dan termasuk dalam satu paket addon module.

4.2 PENEMPATAN FOLDER

By default, modul addon harus ditempatkan dibawah folder `addons` di struktur directory odoo, misalnya kalo odoo kita diinstall di folder `/opt/odoo-10`:

```
/opt/odoo-10/odoo/addons
```

Tapi sebetulnya addons yang kita buat boleh aja disimpan di sembarang folder asalkan folder itu di-set sebagai folder addons yang perlu dicari oleh odoo saat di jalankan.

Caranya adalah dengan nambahin `addons_path` pada file konfigurasi odoo, yaitu `odoo-server.conf`.

```
addons_path = odoo addons/, /users addons/
```

Artinya kita minta odoo untuk nyari addons di folder bawaannya yaitu `odoo/addons` dan juga folder kita yaitu `/users/addons`.

5 BIKIN ADDONS ACADEMIC

Kita asumsikan bikin folder addonsnya di lokasi addons standard odoo, yaitu `odoo/addons`.

5.1 BIKIN FOLDER

Bikin new folder dibawah situ, kasi nama `academic`.

5.2 BIKIN FILE `__OPENERP__.PY`

Bikin file dengan nama `__openerp__.py`

Isinya seperti ini:

```
{  
    "name": "Academic Information System Day 1",  
    "version": "1.0",  
    "depends": [  
        "base",  
    ],  
    "author": "akhmad.daniel@gmail.com",  
    "category": "Education",  
    'website': 'http://www.vitraining.com',  
    "description": """\nAcademic Information System Day 1  
"""",  
    "data": [  
    ],  
    "installable": True,  
    "auto_install": False,  
    "application": True,  
}
```

Gambar 24 Isi file `__openerp__.py` pada awalnya

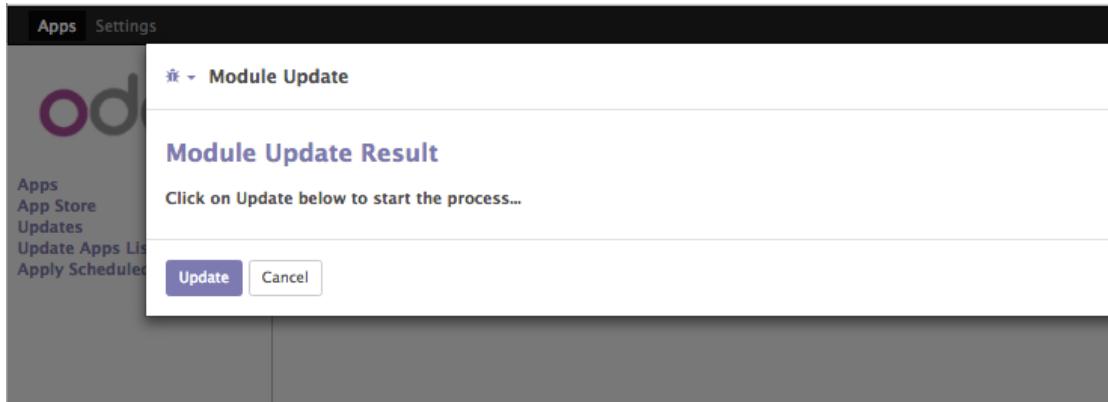
5.3 BIKIN FILE `__INIT__.PY`

Isinya sekarang masih kosong, karena kita belum punya file Python yang mau diimport, yang penting ada dulu filenya.

Struktur folder addons kita harus seperti ini.

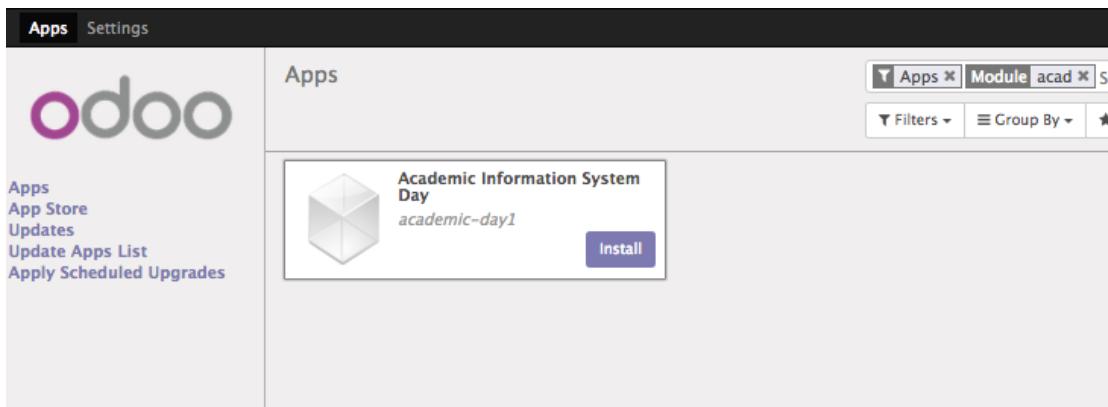
```
|-- odoo  
|   |-- addons  
|       |-- academic  
|           |-- __openerp__.py  
|           \-- __init__.py
```

Supaya odoo kenal modul yang baru kita bikin diatas, harus klik menu **Settings > Apps > Update Apps List**.



Gambar 25 Update Apps List

Kalau proses pencarian modul baru udah selesai, maka modul yang baru kita buat akan terdaftar waktu kita klik menu **Settings > Apps > Apps**. Cari nama module yaitu **academic**, hilangkan filter Installed jika perlu.



Gambar 26 Modul sudah ter-detect oleh odoo

Jangan dulu klik **Install** saat ini.

5.4 RELOAD ODOO

Selama development addons kita perlu restart odoo supaya dia ngeload ulang perubahan pada modul yang baru dilakukan. Sebetulnya ada fasilitas **Upgrade** module di odoo tapi nggak bisa dilakukan sepenuhnya untuk update modul.

Jika ada perubahan di file PY maka perlu restart Odoo.

Jika hanya perubahan di file XML, cukup klik Upgrade module.

5.4.1 LINUX

Di linux gunakan command ini di dalam folder instalasi odoo untuk meng-update modul kita...

```
./odoo-server -c odoo-server.conf --update=academic
```

Pastikan juga kalo odoo udah diinstall secara service, servicenya dimatikan dulu supaya nggak bentrok.

5.4.2 WINDOWS

Development addon module lebih baik dilakukan di Linux, tapi memang bisa juga di Windows.

Untuk meng-update addon module, gunakan juga command --update melalui Command Prompt seperti di Linux...

```
cd C:\program files\odoo\server\odoo  
odoo-server.exe -c odoo-server.conf --update academic
```

Untuk di Windows, pastikan odoo di Windows Service sudah di stop, karena kalo nggak distop bisa menyebabkan dua service odoo jalan, yaitu yang di service dan yang di jalankan manual dari Command Prompt untuk development.

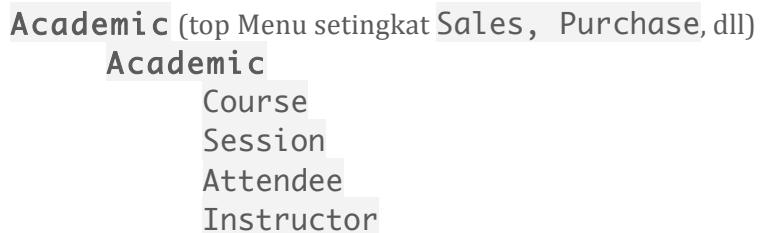
6 BIKIN MENU DAN ACTIONWINDOW

Langkah pertama setelah membuat folder addons `academic` dan file `__openerp__.py` dan `__init__.py` adalah membuat menu aplikasi dan halaman yang muncul ketika menu itu diklik.

6.1 BIKIN MENU

Menu aplikasi yang mau dibuat terdiri dari menu utama dan submenu. Menu utama ditempatkan di manu bar paling atas odoo, dan kalo di klik muncul sub menu di bawahnya.

Susunan menu yang hendak kita rancang kira-kira seperti ini...



Bikin file `menu.xml` dibawah folder `academic`.

Isinya begini...

```
<odoo>
    <data>

        <menuitem id="academic_0"
                  name="Academic"
                  sequence="20"/>

        <menuitem id="academic_1"
                  name="Academic"
                  parent="academic_0"
                  sequence="20"/>

        <menuitem id="menu_academic_course"
                  name="Course"
                  parent="academic_1"
                  action="action_course_list"
                  sequence="20"/>

    </data>
</odoo>
```

Gambar 27 File `menu.xml`

Setiap file XML di odoo berada didalam tag `<odoo>` dan `<data>`. Lalu di setiap `<data>` boleh dimasukkan macam-macam tag

yang tujuannya untuk meng-insert data ke table internal dan semua table lain yang ada di odoo.

Dalam hal ini kita meng-insert tag `<menuitem>` yang gunanya untuk menampilkan menu di odoo.

Tag `<menuitem>` punya banyak attribute, diantaranya..

Attribute `id` adalah referensi untuk setiap record menuitem yang nantinya bisa dipakai sebagai referensi buat menu yang lain.

Attribute `name` gunanya untuk nampilin label menu.

Attribute `action` gunanya untuk nentuin apa nama action window (semacam halaman) yang akan ditampilkan ketika menu ini diklik.

Attribute `sequence` gunanya untuk nentuin urutan munculnya menu ini diantara menu lain yang setingkat.

Attribute `parent` gunanya untuk nentuin induk dari menu ini. Isinya adalah attribute `id` dari menu induk tersebut. Contohnya menu Course, Session, dan Attendee induknya Academic. Sementara menu Academic sendiri induknya adalah menu Adademic yang ada di top menu.

Jadi sesuai rancangan dan XML yang baru dibuat, kira-kira gini nih mapping nya...

```
Academic      (id academic_0)
  Academic    (id academic_1, parent academic_0)
    Course     (id academic_1_1, parent academic_1)
    Session    (belum dibuat)
    Attendee   (belum dibuat)
    Instructor (belum dibuat)
```

6.2 BIKIN ACTION WINDOW

Next, kita perlu buat record action window untuk nangkep klik dari menu. Action window boleh kita ibaratkan kayak halaman web yang muncul waktu suatu link diklik.

Contohnya di XML tadi, waktu kita klik menu `Course`, action diarahin ke `action_course_list`. Berarti kita perlu bikin action window dengan id `action_course_list`.

Penentuan `id` di odoo boleh bebas asal representative dan nggak salah dalam me-referensikan dari elemen lain.

Saat suatu elemen XML direferensikan oleh elemen lain, syaratnya elemen yang direferensikan tersebut harus udah ada di atas yang me-referensikan.

Contohnya `action_course_list` di-refer dari menu `Course`, berarti sebelum menu item `Course` dibuat, harus udah ada dulu action window `action_course_list`. Artinya harus ditulis di bagian atas sebelum menu item.

Jadi edit lagi `menu.xml`, tambahi action window untuk masing-masing menu...

Ini action window untuk Course List...

```
<odoo>
<data>

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

```
<menuitem id="academic_0"
          name="Academic"
          sequence="20"/>

<menuitem id="academic_1"
          name="Academic"
          parent="academic_0"
          sequence="20"/>

<menuitem id="menu_academic_course"
          name="Course"
          parent="academic_1"
          action="action_course_list"
          sequence="20"/>
</data>
</odoo>
```

Gambar 28 menu.xml udah ditambahi action window course

Jangan lupa, tambahan action window harus ditulis di atas deklarasi menu item yang udah kita buat sebelumnya.

Deklarasi action window menggunakan tag `<record>`. Tag ini gunanya untuk menginsert record langsung ke object (table) yang ditulis pada attribute `model`. Attribut lainnya yaitu `id` gunanya sebagai id kalau elemen ini mau di-refer oleh elemen XML lain, contohnya di-refer dari menu.

Waktu memakai elemen tag `<record>` untuk meng-insert record ke suatu object, didalamnya harus ditentukan nilai dari setiap field dari record yang akan diinsert.

Contohnya, waktu kita insert record ke object action window (nama lengkapnya `ir.actions.act_window`), field yang mandatory yang harus disertakan adalah field `name`, `res_model`, dan `view_mode`.

Field `name` adalah nama dari action window yang mau diinsert.

Field `res_model` adalah nama model (object/ table) yang dijadikan sumber data di halaman yang akan tampil. Ini adalah deklarasi model Python dan harus kita buat nanti

Field `view_mode` nentuin gimana data itu akan ditampilkan, apakah dalam list (tree) view, form, graph, calendar, dan

lainnya... Field ini nantinya menentukan icon-icon apa aja yang muncul di list view Course untuk melihat data dalam list atau form view.

Field `help` sih sifatnya nggak mandatory, tapi kalo diisi gunanya untuk munculin informasi ketika belum ada record di table sumber data (`res_model`).

Oh iya, di odoo istilah object mewakili model class, dan setiap model class langsung berhubungan dengan suatu table di database. Jadi misalnya kita buat object `Course` pada module addons `academic`, maka di database otomatis ter-create table dengan nama `academic_course`. Field-fieldnya menyesuaikan dengan model class yang kita buat nanti di Python.

6.3 UPDATE __OPENERP__.PY

Setiap kali nambah suatu file XML, kita perlu cantumin file itu pada file `__openerp__.py` supaya waktu modul diinstall, file XML itu ikut diproses.

Taronya di key `data`, yang berupa list dari file-file XML yang mau disertakan, dipisahkan pake tanda koma kalo udah lebih dari satu file.

Seperti ini caranya...

```
{  
    "name": "Academic Information System Day",  
    "version": "1.0",  
    "depends": [  
        "base",  
    ],  
    "author": "akhmad.daniel@gmail.com",  
    "category": "Education",  
    "website": 'http://www.vitraining.com',  
    "description": """\\n  
Academic Information System Day1  
  
"""",  
    "data": [  
        "menu.xml",  
    ],  
    "installable": True,  
}
```

```
"auto_install": False,  
"application": True,  
}  
}
```

Gambar 29 Cantumin menu.xml di __openerp__.py

Pada step ini modul belum bisa di update karena kita belum buat file class model yang dipanggil sebagai sumber data di action window di atas.

Lanjut ke bagian model class dulu setelah ini...

7 CLASS COURSE

Udah di bahas sebelumnya, action window perlu informasi dari mana sumber data waktu dia dipanggil oleh menu. Ini ditentukan di field `res_model`. Isinya adalah nama model class Python. Format penamaan class ini kita seragamkan saja dengan `nama_addons.nama_class`, walaupun sebetulnya boleh bebas tapi untuk menghindari kebingungan ke depannya.

Nama addons adalah sama dengan nama folder addon module yang udah kita buat sebelumnya, yaitu `academic`.

7.1 BIKIN CLASS COURSE

Pertama-tama buat file baru dibawah folder addon `academic`. Kasi nama yang mewakili object yang ada di dalamnya, yaitu `course.py`.

Isi file ini adalah deklarasi class `course` dan semua atribut dan method yang ada pada class tersebut.

Sesuai ERD dan spesifikasi table, berikut ini syntax deklarasi class `course` pada file `course.py`.

```
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")
```

Gambar 30 Bikin class Course

Disini, class `course` buatan kita merupakan turunan dari class `models.Model` (bawaan odoo/OpenObject). Otomatis dia bakalan udah punya sifat (method dan properties) sama seperti induknya. Contohnya langsung connect ke table `academic_course`, bisa cari data, insert, update, delete, dan sebagainya.

Setiap class harus diset property `_name` dan `_columns` nya.

Property `_name` gunanya sebagai referensi bagi semua modul addons yang ada di odoo. Jadi kalo ada class atau XML lain yang perlu akses ke class ini, dia harus panggil dengan panggilan `academic.course`.

Seperti yang udah dilakukan oleh action view `action_course_list` sebelumnya dimana dia perlu field `res_model`, dan disana kita isikan dengan nama class course, yaitu `academic.course`.

Property `_columns` gunanya untuk definisi kolom class ini. Property ini bentuknya Python dictionary, dimana ada key (nama kolom) dan value (jenis dan atribut kolom). Disini kita definisiin kolom `name`, `description`, dan `responsible_id`.

Untuk mudahnya, ibaratkan aja class Python ini langsung sebagai table database, karena nantinya setiap class pasti ada korelasinya dengan sebuah table di database, yang akan dibuat otomatis oleh odoo waktu kita install module addon ini.

Sesuai ERD, kolom `name` bertipe `char` dengan panjang maximal 100, nggak boleh kosong alias `required=True`, dengan label di semua tampilan adalah `Name`.

Lalu ada kolom `description`, bertipe `text` (lebih dari satu baris) dengan label di tampilan adalah `Description`.

Lalu ada kolom `responsible_id`. Ini typenya khusus yaitu `many2one` ke object lainnya yaitu `res.users`, label di tampilan adalah `Responsible`. Artinya, kolom ini punya relasi ke table lain, yaitu `res.users` (yaitu user yang bisa login di odoo) dan sifatnya `many2one`, artinya banyak course bisa dimiliki oleh satu user, alias satu user bisa punya banyak course. Ini sesuai dengan spesifikasi dan ERD aplikasi kita.

Konvensi: semua kolom berjenis relasi `many2one` kita buat dengan akhiran `_id`, supaya memudahkan kita dalam mengidentifikasi type kolom ini nantinya.

Kalo udah siap, panggil file `course.py` dari file `__init__.py`, supaya class kita diimport oleh odoo waktu dijalankan. Caranya gini...

```
import course
```

Gambar 31 Import class Course

Susunan file pada addons `academic` kita sejauh ini adalah seperti ini... ada tambahan `menu.xml` dan `course.xml`.

```
addons/academic
|-- __init__.py
|-- __openerp__.py
|-- course.py
\-- menu.xml
```

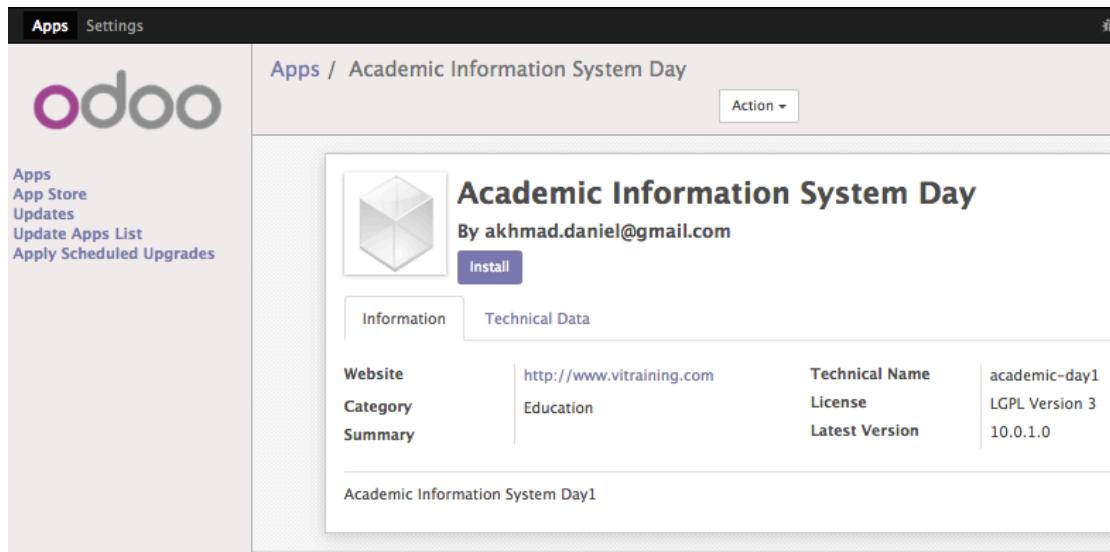
Mari test tampilan aplikasi kita...

Restart `odoo-server` dan update modul addon `academic`.

```
./odoo-server -c odoo-server.conf --u academic
```

Install module addons `academic`.

Cari modul dari menu **Apps > Apps**.

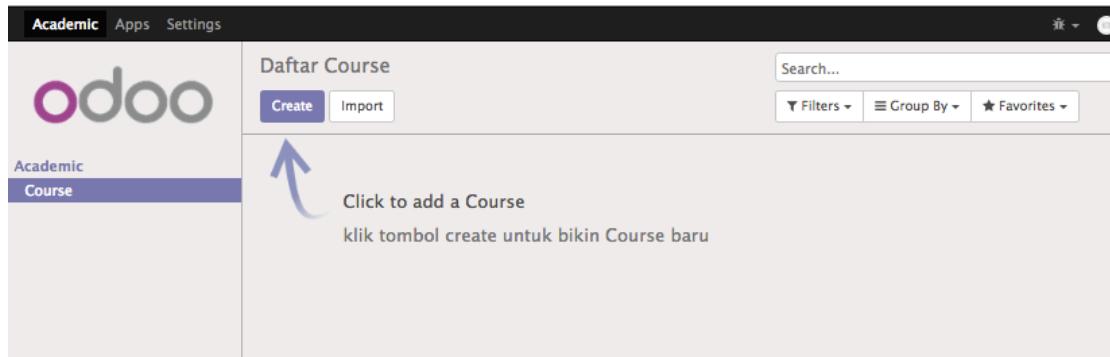


Gambar 32 Install module academic

Klik **Install**.

Lihat hasilnya di web browser... klik menu **Academic** di top menu.

Muncul list view dari course dimana belum ada data Course dan muncul tulisan help.



Gambar 33 Menu course udah muncul

Klik tombol **Create**...

Muncul form view course dimana kita bisa isikan data Course. Kolom-kolomnya udah sesuai dengan yang kita definisikan di class course.

Kolom **Name** berupa text box dengan panjang maximal 100 dan nggak boleh kosong. Coba dikosongin terus klik tombol Save, nanti ada validasi bahwa dia nggak boleh kosong.

Kolom **Description** udah berupa text box beberapa baris dan boleh kosong.

Kolom **Responsible** udah berupa pilihan drop down list karena kita set relasi many2one ke object `res.users`.

The screenshot shows the Odoo interface for creating a new course. The top navigation bar includes 'Academic', 'Apps', and 'Settings'. On the left, a sidebar shows 'odoo' and 'Academic Course'. The main window title is 'Daftar Course / New'. It contains three input fields: 'Description' (a text area with placeholder 'Description'), 'Responsible' (a dropdown menu with placeholder 'Responsible'), and 'Name' (a text area with placeholder 'Name'). At the top right are 'Save' and 'Discard' buttons. The status bar at the bottom right shows 'Administrator (aca)'.

Gambar 34 Sudah bisa create Course

Klik tombol **Save**. Akan terlihat tampilan list view data Course yang udah masuk. Dari list view boleh di klik untuk masuk lagi ke form view.

Boleh juga di centang untuk proses selanjutnya seperti **Delete**.

The screenshot shows the Odoo interface for viewing a list of courses. The top navigation bar includes 'Academic', 'Apps', and 'Settings'. On the left, a sidebar shows 'odoo' and 'Academic Course'. The main window title is 'Daftar Course'. It displays a list of courses with columns for 'Name' and 'PHP'. At the top right are buttons for 'Search...', 'Filters', 'Group By', and 'Favorites'. The status bar at the bottom right shows 'Administrator (aca)'.

Gambar 35 Sudah bisa lihat list Course yang udah tersimpan dan bisa diedit dan delete

By default, odoo udah nampilin object yang kita mau di list view dan form view dengan susunan standard, yaitu hanya beberapa

field yang muncul di list view dan susunan field di form view masih apa adanya.

7.2 BIKIN TREE VIEW COURSE

Tampilan list view Course boleh kita modif supaya muncul semua field yang ada di class **Course**.

Caranya, bikin file baru dibawah addons **academic**, kasi nama **course.xml**. Disini nanti kita akan simpan semua definisi yang berkaitan dengan view Course. Salah satunya tampilan list view yang mau kita modif.

Isinya adalah...

```
<?xml version="1.0" encoding="utf-8"?>
<openerp>
    <data>

        <record id="view_academic_course_tree" model="ir.ui.view">
            <field name="name">academic.course.tree</field>
            <field name="model">academic.course</field>
            <field name="type">tree</field>
            <field name="priority" eval="8"/>
            <field name="arch" type="xml">
                <tree string="course">
                    <field name="name"/>
                    <field name="description"/>
                    <field name="responsible_id"/>
                </tree>
            </field>
        </record>

    </data>
</openerp>
```

Gambar 36 Definisi tree view Course

Untuk memodif tampilan list view suatu object, di odoo kita perlu buat definisi tree view di XML terhadap object tersebut.

Contoh di atas kita bikin definisi tree view untuk object **academic.course**, yaitu dengan cara membuat record baru pada object **ir.ui.view** yang terdefinisi pada atribut **model**.

Atribut `id` sama seperti elemen record XML yang sebelumnya, merupakan referensi bagi elemen XML lainnya.

Record ini secara internal berkaitan dengan tampilan suatu object tergantung dari definisi arsitekturnya nanti.

Record ini perlu field berikut.

Field `name` adalah nama internal tree view ini.

Field `model` adalah nama object model yang mau didefinisikan tampilan list tree view nya.

Field `arch` menentukan bagaimana object ini ditampilkan, apakah secara tree, form, calendar, dan lain sebagainya.

Disini kita buat definisi arsitektur dalam tree view, caranya dengan membuat definisi tag `tree` di dalam tag field `arch`.

Di dalam tag `tree`, kita tinggal panggil nama-nama field yang mau dimunculin di list tree view. Otomatis karena kita ngomongnya object course, berarti semua field yang ada di object tersebut bisa dimunculkan disini.

Contoh disini kita munculin field `name`, `description`, dan `responsible_id`.

Lanjut...

Tambahkan file `course.xml` pada file `__openerp__.py`...

```
{  
    "name": "Academic Information System Day",  
    "version": "1.0",  
    "depends": [  
        "base",  
    ],  
    "author": "akhmad.daniel@gmail.com",  
    "category": "Education",  
    'website': 'http://www.vitraining.com',  
    "description": """\nAcademic Information System Day1
```

```

    """
    "data": [
        "menu.xml",
        "course.xml"
    ],
    "installable": True,
    "auto_install": False,
    "application": True,
}

```

Gambar 37 Include course.xml di __openerp__.py

Susunan file pada addons `academic` kita sejauh ini adalah seperti ini... ada tambahan file `course.xml`.

```

addons/academic
|-- __init__.py
|-- __openerp__.py
|-- course.py
|-- course.xml
`-- menu.xml

```

Lanjut...

Upgrade module, coba lihat tampilan list view Course yang baru...

Daftar Course		
		Search...
	Name	Description
<input type="checkbox"/>	PHP	kursus PHP

Gambar 38 Course list udah muncul

okee... udah muncul field nya sesuai kemauan kita...

yang menarik disini, kolom `Responsible` udah langsung muncul nama user yang direlasikan ke Course.

7.3 BIKIN FORM VIEW COURSE

Lanjut, kita mau modif juga tampilan form view Course supaya lebih bagus, nggak standard kayak yang sekarang ada.

Edit lagi file course.xml.

Tambahi deklarasi form view untuk class Course seperti ini...

```
<?xml version="1.0" encoding="utf-8"?>
<openerp>
    <data>

        <record id="view_academic_course_tree" model="ir.ui.view">
            <field name="name">academic.course.tree</field>
            <field name="model">academic.course</field>
            <field name="type">tree</field>
            <field name="priority" eval="8"/>
            <field name="arch" type="xml">
                <tree string="course">
                    <field name="name"/>
                    <field name="description"/>
                    <field name="responsible_id"/>
                </tree>
            </field>
        </record>

        <record id="view_academic_course_form" model="ir.ui.view">
            <field name="name">academic.course.form</field>
            <field name="model">academic.course</field>
            <field name="type">form</field>
            <field name="priority" eval="8"/>
            <field name="arch" type="xml">
                <form string="Course Form">

                    <sheet>
                        <div class="oe_title">
                            <h1>
                                <field name="name"/>
                            </h1>
                        </div>
                        <group>
                            <field name="responsible_id"/>
                        </group>
                    </sheet>
                    <notebook>
                        <page string="Description">
                            <field name="description"/>
                        </page>
                    </notebook>
                </form>
            </field>
        </record>

    </data>
</openerp>
```

Gambar 39 Definisi form view Course

Sama seperti tree view, definisi form view juga memakai record object model `ir.ui.view`, karena terkait dengan tampilan.

Bedanya disini adalah definisi field `arch` nya.

Disini dipake tag `form` yang ditaro didalam field `arch`, yang gunanya untuk mendefinisikan ulang tampilan form dari object yang sedang diomongin yaitu `academic.course`.

Pertama-tama atribut `string` tag ini yang gunanya untuk label judul form.

Lalu atribut `version` yang nentuin jenis versi tag form apakah versi 10 atau bukan. Versi 10 punya lebih banyak fitur misalnya boleh pasang tag HTML dan CSS didalamnya.

Didalam tag form ada tag `sheet` tempat kita naro semua elemen field yang ada pada object model di atas form.

Pertama kita taro field `name` yang udah kita format pake tag HTML `div` dan `h1`... supaya tampilan font nya muncul lebih besar.

Lalu ada tag `group` yang gunanya untuk ngelompokin field-field dalam satu kelompok dan memunculkan label. Field yang di luar tag `group` nggak muncul labelnya.

Lanjut, ada tag `notebook` didalam tag `form`. Ini untuk membuat tempat untuk tab-tab yang bisa berisi field-field lain supaya tampilan lebih rapih dan terstruktur...

Didalam tag `notebook` dicantumin tag `page`. Setiap `page` akan jadi satu tab nantinya. Disini kita baru bikin satu tag `page` yang isinya field `description`.

Lanjut... restart server dan update module.

Lihat hasilnya...

The screenshot shows a web-based form titled "PHP". At the top left are "Save" and "Discard" buttons. Top right shows "1 / 1" and navigation icons. The main area has a title "PHP" in a large font. Below it, a "Responsible" field is set to "Administrator". A "Description" field contains the text "kursus PHP".

Gambar 40 Form view Course

okee,... tampilan form view udah berubah, ada tab **Description** dan nama field Course jadi besar font-nya.

Tapi tunggu... waktu di edit gimana caranya supaya label field name muncul? Hanya waktu edit aja, kalau view sih nggak apa...

Edit lagi file `course.xml`, tambahi ini...

```
<record id="view_academic_course_form" model="ir.ui.view">
    <field name="name">academic.course.form</field>
    <field name="model">academic.course</field>
    <field name="type">form</field>
    <field name="priority" eval="8"/>
    <field name="arch" type="xml">
        <form string="Course Form">
            <sheet>
                <div class="oe_title">
                    <label for="name" class="oe_edit_only"
                           string="Course Name"/>
                    <h1>
                        <field name="name"/>
                    </h1>
                </div>
            </sheet>
        </form>
    </field>
</record>
```

Gambar 41 Label Course muncul hanya pada saat edit

Disini kita tambahi tag HTML `label` untuk dan sebelum field `name` tapi dikasi class CSS khusus `oe_edit_only` yang membuat label itu hanya muncul waktu form edit.

Restart lagi server dan update module...

Harusnya label edit udah muncul.

Daftar Course / PHP

Save Discard 1 / 1 <

Course Name	PHP
Responsible	Administrator
Description	kursus PHP

Beres sementara untuk urusan Course, kita lanjut ke urusan object Session...

8 CLASS SESSION

8.1 MENU DAN ACTION WINDOW SESSION

Agar supaya bisa mengakses object Session kita perlu bikin dulu:

- menu **Session List** yang memanggil action window
- action window yang memanggil class Session
- class Session

Pertama kita buat dulu menu untuk Session. Edit file `menu.xml`.

Tambahi tag `menuitem` dibawah deklarasi menu Course.

```
<menuitem id="menu_academic_course"
          name="Course"
          parent="academic_1"
          action="action_course_list"
          sequence="20"/>

<menuitem id="menu_academic_session"
          name="Session"
          parent="academic_1"
          action="action_session_list"
          sequence="30"/>
</data>
</odoo>
```

Gambar 42 Menu Session

Intinya kita nambahi tag `menuitem` dengan attribut `parent` adalah menu Academic (id `academic_1`) ...

dan `action` adalah action window dengan id `action_session_list`.

Udah itu, buat action window yang dimaksud untuk Session List...

```
<odoo>
  <data>

    <record id="action_course_list" model="ir.actions.act_window">
      <field name="name">Daftar Course</field>
      <field name="res_model">academic.course</field>
      <field name="view_mode">tree,form</field>
      <field name="help" type="html">
        <p class="oe_view_nocontent_create">
```

```

        Click to add a Course
    </p>
    <p>klik tombol create untuk bikin Course baru</p>
</field>
</record>

<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</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>

<menuitem id="academic_0"
    name="Academic"
    sequence="20"/>

```

Gambar 43 Action window Session

Baris ini harus diatas deklarasi `menuitem` Session List...

Ini sama aja seperti waktu kita mendeklarasikan action window untuk Course List, hanya saja disini kita kasi atribut `id = action_session_list` sesuai dengan yang di-refer pada menu Session List ...

dan link ke object `academic.session` sebagai sumber datanya yang didefinikan pada field `res_model`.

Susunan menu kita sejauh ini adalah...

```

Academic      (id academic_0)
    Academic     (id academic_1, parent academic_0)
        Course   (id academic_1_1, parent academic_1)
        Session  (id academic_1_2, parent academic_1)
        Attendee (belum dibuat)
        Instructor (belum dibuat)

```

8.2 BIKIN CLASS SESSION

Action window session list perlu class object sebagai sumber datanya yaitu dengan nama `academic.session`.

Kita buat class itu sekarang.

Buat file baru dibawah folder addon `academic`. Kasi nama yang mewakili object yang ada di dalamnya, yaitu `session.py`.

Isi file ini adalah deklarasi class `session` dan semua atribut dan method yang ada pada class tersebut.

Sesuai ERD dan spesifikasi table, berikut ini syntax deklarasi class `session` pada file `session.py`.

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

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

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

    course_id = fields.Many2one(comodel_name="academic.course",
        string="Course", required=True, )
    instructor_id = fields.Many2one(comodel_name="res.partner",
        string="Instructor", required=True, )
    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", )
```

Gambar 44 Session Class

Sama seperti class Course sebelumnya, class `session` buatan kita merupakan turunan dari class `models.Model` (bawaan odoo/OpenObject). Otomatis dia bakalan udah punya sifat (method dan properties) sama seperti induknya. Contohnya langsung connect ke table `academic_session`, bisa cari data, insert, update, delete, dan sebagainya.

Setiap class harus diset property `_name` dan `_columns` nya.

Property `_name` gunanya sebagai referensi bagi semua modul addons yang ada di odoo. Jadi kalo ada class atau XML lain yang perlu akses ke class ini, dia harus panggil dengan panggilan `academic.session`.

Property `_columns` gunanya untuk definisi kolom class, yaitu kolom `name`, `course_id`, dan `instructor_id`, `start_date`, `duration`, `seats`, dan `active` sesuai dengan definisi ERD.

Kalo udah siap, panggil file `session.py` dari file `__init__.py`, supaya class kita diimport oleh odoo waktu dijalankan. Caranya gini...

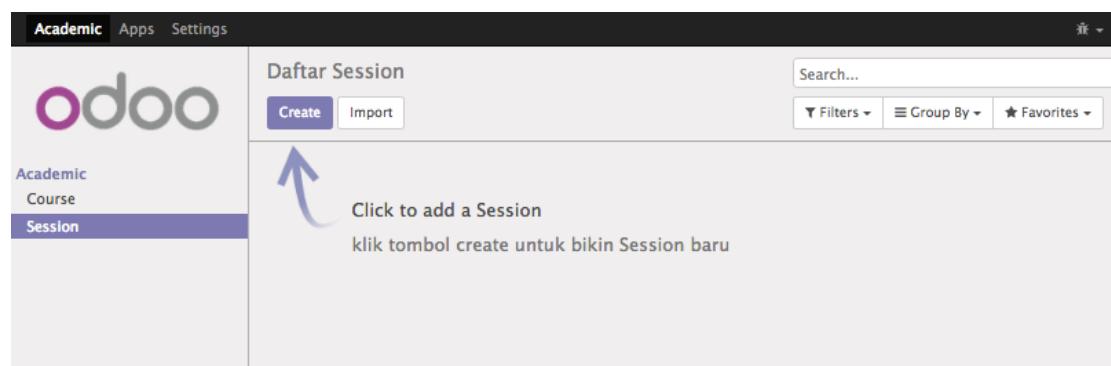
```
import session
import course
```

Gambar 45 Import Session

Susunan file pada addons `academic` kita sejauh ini adalah seperti ini... ada tambahan `session.py`.

```
addons/academic
|-- __init__.py
|-- __openerp__.py
|-- session.py
|-- course.py
`-- menu.xml
```

Yuuk,... restart odoo dan update module.. mari test tampilan aplikasi kita sekarang, jangan sampe ada error 😊 ...



Gambar 46 Menu Session muncul

okee sekarang udah nongol menu **Session** dan udah bisa klik tombol **Create**...

Daftar Session / New

Save **Discard**

Name	<input type="text"/>	Instructir	<input type="text"/>
Duration	<input type="text"/> 0	Seats	<input type="text"/> 0
Course	<input type="text"/>	Active	<input type="checkbox"/>
Start Date	<input type="text"/>		

Gambar 47 Bisa add session

klik tombol Save...

Daftar Session / Session 1

Edit **Create** **Action** **1 / 1**

Name	Session 1	Instructir	
Duration	0	Seats	0
Course	PHP	Active	<input type="checkbox"/>
Start Date			

Gambar 48 Bisa save dan view Session

klik List View icon atau menu Session...

Daftar Session

Create **Import** **Active is false** **Search...**

Filters **Group By** **Favorites**

Active is false

Add Custom Filter

<input type="checkbox"/> Name	
<input type="checkbox"/> Session 1	

Gambar 49 Tree view Session

dan **Edit**, **Delete** Session... semua udah berfungsi.

Penting: Kolom **active** punya arti khusus di Odoo, yaitu jika isinya false, maka record nggak muncul di list dan form view, seolah-olah nggak ada data tersebut. Untuk membuka record yang nggak active, klik **Add Custom Filter**, pilih kolom **Active** is False lalu klik **Apply**.

The screenshot shows the Odoo interface for managing sessions. On the left, there's a list of sessions with one entry named 'Session 1'. On the right, there's a search bar and a filter panel. The filter panel has a dropdown for 'Active' set to 'is false', and an 'Apply' button is highlighted with a blue border. Other buttons include 'Add Custom Filter' and 'Add a condition'.

8.3 MODIF LIST VIEW SESSION

Modif tampilan list view Session supaya muncul field-field yang kita perlukan.

Caranya, bikin file baru dibawah addons `academic`, kasi nama `session.xml`. Disini nanti kita akan simpan semua definisi yang berkaitan dengan view Session. Salah satunya tampilan list view yang mau kita modif.

Isinya adalah...

```
<?xml version="1.0" encoding="utf-8"?>
<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="name"/>
                    <field name="instructor_id" />
                    <field name="start_date" />
                    <field name="duration" />
                </tree>
            </field>
        </record>
    </data>
</openerp>
```

```
        <field name="seats" />
        <field name="active" />
    </tree>
</field>
</record>

</data>
</openerp>
```

Gambar 50 Modif list view Session

Sama seperti waktu Course List, untuk memodif tampilan list view suatu object, di odoo kita perlu buat definisi tree view di XML terhadap object tersebut.

Disini kita bikin definisi tree view untuk object `academic.session`. Caranya dengan membuat record baru pada object `ir.ui.view` yang atribut `model` nya adalah object yang mau didefinisikan tree view nya.

Atribut `id` sama seperti elemen record XML yang sebelumnya, merupakan referensi bagi elemen XML lainnya.

Record ini perlu field berikut.

Field `name` adalah nama internal tree view ini.

Field `model` adalah nama object model yang mau didefinisikan tampilan list tree view nya.

Field `arch` menentukan bagaimana object ini ditampilkan, apakah secara tree, form, calendar, dan lain sebagainya.

Disini kita buat definisi arsitektur dalam tree view, caranya dengan membuat definisi tag `tree` di dalam tag field `arch`.

Di dalam tag `tree`, kita panggil nama-nama field yang mau dimunculin di list tree view. Otomatis karena kita sekarang ngomongnya object session, berarti semua field yang ada di object tersebut bisa dimunculkan disini.

Contoh disini kita munculin semua field yang ada pada object Session.

Lanjut...

Tambahkan file `session.xml` pada file `__openerp__.py`...

```
{  
    "name": "Academic Information System Day",  
    "version": "1.0",  
    "depends": [  
        "base",  
    ],  
    "author": "akhmad.daniel@gmail.com",  
    "category": "Education",  
    "website": 'http://www.vitraining.com',  
    "description": """\nAcademic Information System Day1  
"""",  
    "data": [  
        "menu.xml",  
        "course.xml",  
        "session.xml",  
    ],  
    "installable": True,  
    "auto_install": False,  
    "application": True,  
}
```

Gambar 51 Panggil `session.xml` dari `__openerp__.py`

Susunan file pada addons `academic` kita sejauh ini adalah seperti ini... ada tambahan file `session.xml`.

```
addons/academic  
|-- __init__.py  
|-- __openerp__.py  
|-- course.py  
|-- course.xml  
|-- session.xml  
`-- menu.xml
```

Lanjut...

Restart server, coba lihat tampilan list view Session yang baru...

Daftar Session					
	Name	Instructir	Start Date	Duration	Seats
	Session 1	Administrator	02/19/2017		21 <input checked="" type="checkbox"/>

Gambar 52 List view Session udah lengkap

okee... udah muncul field nya sesuai kemauan kita...

Perhatikan kolom **Instructor** udah langsung muncul nama **Partner** yang direlasikan ke Session.

Kolom **Is Active** juga sudah muncul checkbox sesuai definisi di model class.

8.4 MODIF FORM VIEW

Sekarang kita modif form view Session supaya tampilan lebih terstruktur.

Edit lagi file **session.xml**.

Tambahi deklarasi form view untuk class Session seperti ini...

```
<?xml version="1.0" encoding="utf-8"?>
<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="name"/>
                    <field name="course_id" />
                    <field name="instructor_id" />
                    <field name="start_date" />
                    <field name="duration" />
                    <field name="seats" />
                    <field name="active" />
                </tree>
            </field>
        </record>

        <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" />
                    <field name="start_date" />
                </group>
                <group>
                    <field name="duration" />
                    <field name="seats" />
                    <field name="active" />
                </group>
            </group>
        </sheet>
    </form>
</field>
</record>

</data>
</openerp>

```

Gambar 53 Modif form view Session

Sama seperti from view Course sebelumnya, definisi form view memakai record object model `ir.ui.view` dengan tag `form` yang ditaruh didalam field `arch`, yang gunanya untuk mendefinisikan ulang tampilan form dari object yang sedang diomongin yaitu `academic.session`.

Pertama-tama atribut `string` tag ini yang gunanya untuk label judul form.

Lalu atribut `version` yang nentuin jenis versi tag form apakah versi 10 atau bukan. Versi 10 punya lebih banyak fitur misalnya boleh pasang tag HTML dan CSS didalamnya.

Didalam tag form ada tag `sheet` tempat kita naro semua elemen field yang ada pada object model di atas form.

Pertama kita taro field `name` yang udah kita format pake tag HTML `div` dan `h1`... supaya tampilan font nya muncul lebih besar. Lalu ada tag `label` HTML untuk menampilkan label Name hanya pada saat edit aja.

Lalu ada tag `group` yang gunanya untuk ngelompokin field-field dalam satu kelompok dan memunculkan label. Field yang di luar tag `group` nggak muncul labelnya.

Tapi disini group-nya dua level, ada group level satu yang terdiri dari 2 group yang masing-masing berisi field object session. Ini untuk membuat tampilan field terbagi menjadi 2 kolom terpisah.

Lanjut... restart server dan update module.

Lihat hasilnya...

The screenshot shows a web-based form for viewing a session. At the top, there are buttons for 'Save' and 'Discard', and a page navigation indicator '1 / 1'. The main area contains a large input field labeled 'Session Name' with the value 'Session 1'. Below it, there are several form fields grouped into two columns:

Course	Duration
PHP	21
Instructor	Seats
Administrator	122
Start Date	Active
02/19/2017	<input checked="" type="checkbox"/>

Gambar 54 Form view Session

okee,... tampilan form view udah berubah, field udah terkelompok jadi 2 kolom dan nama field Session jadi besar font-nya.

8.5 BIKIN RELASI COURSE HAS MANY SESSION

Sesuai ERD sebelumnya, satu **Course** bisa punya banyak **Session**. Jadi kita perlu edit definisi class model **Course** supaya mendukung relasi ini.

Edit file `course.py`.

Tambahi field baru namanya `session_ids`.

```
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")
```

Gambar 55 Tambah field `session_ids` di Session class

Disini kita definisikan kolom baru dengan nama `session_ids`.

Penamaan ini hanya konvesi aja, akhiran `_id` artinya dia merupakan kolom relasi dan penambahan huruf "s" sehingga menjadi `_ids` artinya relasi ke banyak record pada table lawan.

Kolom ini punya type khusus, `one2many`. Type kolom ini minta parameter sebagai berikut:

Parameter pertama comodel_name, adalah object model lawannya, yaitu yang boleh dimiliki lebih dari satu oleh object ini, disini `academic.session`. Artinya satu record di object `course` boleh punya banyak pasangan record di object `session`.

Parameter kedua inverse_name, adalah nama field pada object lawan, disini `course_id`. Lihat definisi kolom di object `session`, disana ada kolom namanya `course_id` yang menandakan bahwa dia dimiliki oleh suatu record di object `course`.

Parameter ketiga adalah label waktu ditampilkan di view.

Parameter keempat dan seterusnya sifatnya optional, disini kita set `ondelete="cascade"` artinya kalo record pada object `course` dihapus maka record-record terkait pada object `session` juga ikut terhapus.

Setelah dideklarasikan seperti diatas, maka object `course` udah punya satu kolom tambahan dengan type `one2many` dan udah bisa ditampilkan di view seperti halnya field-field regular lainnya.

8.6 MODIF VIEW FORM COURSE – TAMBAH SESSION

Untuk nampilin kolom tambahan pada object `course`, kita modif file `course.xml` yang merupakan kumpulan view untuk object ini.

Tambahkan baris ini pada form view Course...

```
<sheet>
    <div class="oe_title">
        <label for="name" class="oe_edit_only" string="Course Name"/>
        <h1>
            <field name="name"/>
        </h1>
    </div>
    <group>
        <field name="responsible_id"/>
    </group>

    <notebook>
        <page string="Description">
            <field name="description"/>
        </page>
        <page string="Sessions">
            <field name="session_ids">
                <tree string="Sessions">
                    <field name="name"/>
                    <field name="instructor_id" />
                    <field name="start_date" />
                    <field name="duration" />
                    <field name="seats" />
                    <field name="active" />
                </tree>
            </field>
        </page>
    </notebook>
</sheet>
```

Gambar 56 Tambah tab Session di Course form view

Disini kita modif form view Course, yaitu nambahin satu tag `page` baru di dalam tag `notebook` yang udah ada sebelumnya.

Didalam tag `page` kita masukkan tag `field` untuk kolom `session_ids` milik object `course`.

Didalam tag `field` kita pasang tag `tree`.

Disini tag `tree` langsung berkorelasi dengan object yang merupakan relasi dari kolom `session_ids` yaitu `session`.

Artinya kita bisa munculkan kolom-kolom yang ada pada object `session` di dalam tag `tree`. Disini kita munculkan semua kolom yang ada di `session`.

Okee... restart server dan update module.

Lihat hasilnya...

The screenshot shows a web-based form titled 'Daftar Course / PHP'. At the top, there are 'Save' and 'Discard' buttons, and a page number '1 / 1' with navigation arrows. Below the title, there are fields for 'Course Name' (containing 'PHP') and 'Responsible' (set to 'Administrator'). A tabular section labeled 'Sessions' is displayed, containing one row of data:

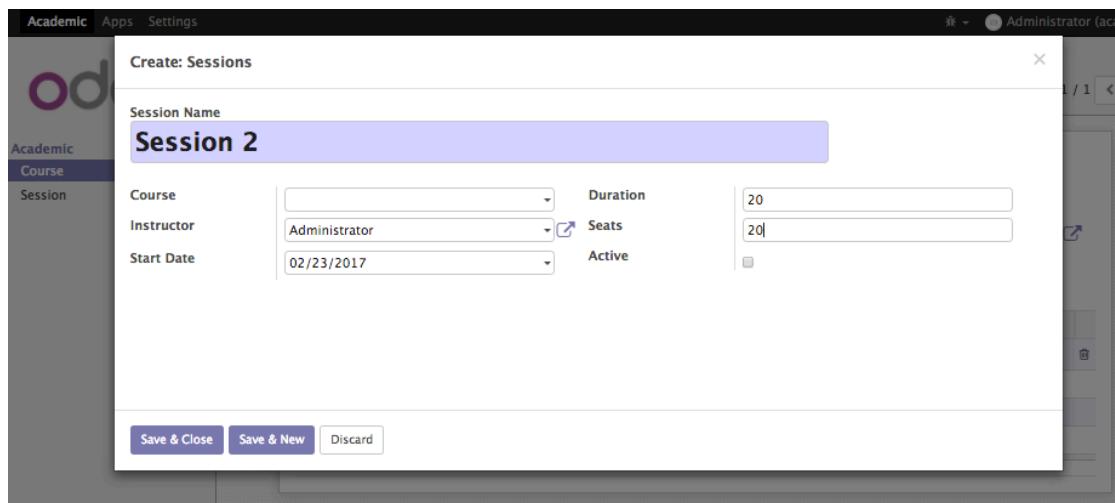
Name	Instructor	Start Date	Duration	Seats	Active
Session 1	Administrator	02/19/2017	21	122	<input checked="" type="checkbox"/>

Below the table, there is a link 'Add an item' and a large empty input field.

Gambar 57 Tab Session muncul di Course form view

Sekarang form Course udah ada tambahan tab baru yang berisi Session yang terkait dengan Course tersebut.

Dan kalau di edit, maka kita bisa tambahkan Session baru langsung dari form Course, dengan klik link `Add new item`.



Gambar 58 Create Session dari Course form view

Form Session yang muncul disini sama persis dengan ketika kita add Session melalui menu Session.

Jangan lupa tick field **Is Active** disini agar record Session tampil.

Tapi Course boleh dikosongkan disini dan otomatis link dengan Course yang sedang di-edit sekarang.

9 CLASS ATTENDEE

9.1 MENU DAN ACTION WINDOW ATTENDEE

Agar supaya bisa mengakses object **Attendee** kita perlu bikin dulu:

- menu **Attendee List** yang memanggil action window
- action window yang memanggil class **Attendee**
- class **Attendee**

Pertama kita buat dulu menu untuk **Attendee**.

Edit file `menu.xml`.

Tambahi tag `menuitem` dibawah deklarasi menu **Course**.

```
<menuitem id="menu_academic_session"
          name="Session"
          parent="academic_1"
          action="action_session_list"
          sequence="30"/>

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

Gambar 59 Tambah menu Attendee

Intinya kita nambahi tag `menuitem` dengan attribut `parent` adalah menu Academic (id `academic_1`) ...

dan `action` adalah action window dengan id `action_attendee_list`.

Udah itu, buat action window yang dimaksud untuk Attendee 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</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>

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

<menuitem id="academic_0"
          name="Academic"
          sequence="20"/>

```

Gambar 60 Action window attendee

Baris ini harus diatas deklarasi menuitem Attendee List...

Ini sama aja seperti waktu kita mendeklarasikan action window untuk Session List, hanya saja disini kita kasi atribut `id = action_attendee_list` sesuai dengan yang di-refer pada menu Attendee List ...

dan link ke object `academic.attendee` sebagai sumber datanya yang didefinikan pada field `res_model`.

Susunan menu kita sejauh ini adalah...

```

Academic      (id academic_0)
    Academic      (id academic_1, parent academic_0)
        Course   (id academic_1_1, parent academic_1)

```

```
Session (id academic_1_2, parent academic_1)
Attendee(id academic_1_3, parent academic_1)
Instructor (belum dibuat)
```

9.2 BIKIN CLASS ATTENDEE

Action window session list perlu class object sebagai sumber datanya yaitu dengan nama `academic.attendee`.

Kita buat class itu sekarang.

Buat file baru dibawah folder addon `academic`. Kasi nama yang mewakili object yang ada di dalamnya, yaitu `attendee.py`.

Isi file ini adalah deklarasi class `attendee` dan semua atribut dan method yang ada pada class tersebut.

Sesuai ERD dan spesifikasi table, berikut ini syntax deklarasi class `attendee` pada file `attendee.py`.

```
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, )
```

Gambar 61 Attendee Class

Sama seperti class Session sebelumnya, class `attendee` buatan kita merupakan turunan dari class `models.Model` (bawaan odoo/OpenObject). Otomatis dia juga bakalan udah punya sifat (method dan properties) sama seperti induknya. Contohnya langsung connect ke table `academic_attendee`, bisa cari data, insert, update, delete, dan sebagainya.

Setiap class harus diset property `_name` dan `_columns` nya.

Property `_name` gunanya sebagai referensi bagi semua modul addons yang ada di odoo. Jadi kalo ada class atau XML lain yang perlu akses ke class ini, dia harus panggil dengan panggilan `academic.attendee`.

Property `_columns` gunanya untuk definisi kolom class, yaitu kolom `name`, `session_id`, dan `partner_id` sesuai dengan definisi ERD.

Kolom `name` bertipe char panjang maximum 200 karakter.

Kolom `session_id` bertipe many2one yaitu relasi ke object `academic.session`.

Kolom `partner_id` bertipe many2one yaitu relasi ke object `res.partner` bawaan odoo.

Kalo udah siap, panggil file `attendee.py` dari file `__init__.py`, supaya class kita diimport oleh odoo waktu dijalankan. Caranya gini...

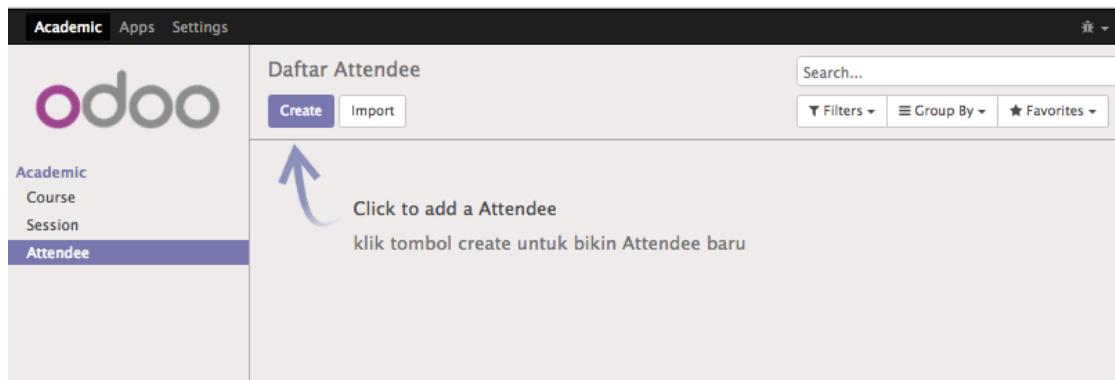
```
import session
import course
import attendee
```

Gambar 62 Import Attendee

Susunan file pada addons `academic` kita sejauh ini adalah seperti ini... ada tambahan `attendee.py`.

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

Lanjut,... restart odoo dan update module.. mari test tampilan aplikasi kita sekarang, jangan sampe ada error 😊 ...



Gambar 63 Muncul Attendee menu

Okee , menu Attendee udah muncul, bisa klik tombol Create...

A screenshot of the 'Daftar Attendee / New' form. At the top, there are 'Save' and 'Discard' buttons. The form fields include 'Name' (containing 'attende'), 'Session' (set to 'Session 1'), and 'Partner' (set to 'Administrator').

Gambar 64 Bisa Add Attendee

Input field-field sesuai definisi model, terutama field Partner dan Session yang berupa many2one.

Bisa klik tombol Save...

Daftar Attendee / attendee			
Edit		Create	Action ▾
Name	attendee	Session	Session 1
Partner	Administrator		

Gambar 65 View Attendee

Ada link yang bisa diklik ke object yang terkait untuk semua field dengan type many2one.

Muncul daftar Attendee ketika di klik dari menu atau icon list view.

Daftar Attendee		Search...	Filters ▾	Group By ▾	Favorites ▾	1-1 / 1
	Name					
<input type="checkbox"/>	attendee					

Gambar 66 List view Attendee

9.3 MODIF LIST VIEW ATTENDEE

Modif tampilan list view Attendee supaya muncul field-field yang kita perlukan.

Caranya, sama seperti yang lain, bikin file baru dibawah addons `academic`, kasi nama `attendee.xml`. Disini nanti kita akan simpan semua definisi yang berkaitan dengan view Attendee. Salah satunya tampilan list view yang mau kita modif.

Isinya adalah...

```
<?xml version="1.0" encoding="utf-8"?>
<openerp>
    <data>
```

```

<record id="view_academic_attendee_tree" model="ir.ui.view">
    <field name="name">academic.attendee.tree</field>
    <field name="model">academic.attendee</field>
    <field name="type">tree</field>
    <field name="priority" eval="8"/>
    <field name="arch" type="xml">
        <tree string="Attendee">
            <field name="name"/>
            <field name="session_id" />
            <field name="partner_id" />
        </tree>
    </field>
</record>

</data>
</openerp>

```

Gambar 67 Modif list view

Sama seperti waktu Session List, untuk memodif tampilan list view suatu object, di odoo kita perlu buat definisi tree view di XML terhadap object tersebut.

Disini kita bikin definisi tree view untuk object `academic.attendee`. Caranya dengan membuat record baru pada object `ir.ui.view` yang atribut `model` nya adalah object yang mau didefinisikan tree view nya.

Atribut `id` sama seperti elemen record XML yang sebelumnya, merupakan referensi bagi elemen XML lainnya.

Record ini perlu field berikut.

Field `name` adalah nama internal tree view ini.

Field `model` adalah nama object model yang mau didefinisikan tampilan list tree view nya.

Field `arch` menentukan bagaimana object ini ditampilkan, apakah secara tree, form, calendar, dan lain sebagainya.

Disini kita buat definisi arsitektur dalam tree view, caranya dengan membuat definisi tag `tree` di dalam tag field `arch`.

Di dalam tag `tree`, kita panggil nama-nama field yang mau dimunculin di list tree view. Otomatis karena kita sekarang ngomongnya object `attendee`, berarti semua field yang ada di object tersebut bisa dimunculkan disini.

Contoh disini kita munculin semua field yang ada pada object Attendee.

Lanjut...

Tambahkan file `attendee.xml` pada file `__openerp__.py`...

```
{  
    "name": "Academic Information System Day",  
    "version": "1.0",  
    "depends": [  
        "base",  
    ],  
    "author": "akhmad.daniel@gmail.com",  
    "category": "Education",  
    "website": 'http://www.vitraining.com',  
    "description": """\nAcademic Information System Day1  
"""",  
    "data": [  
        "menu.xml",  
        "course.xml",  
        "session.xml",  
        "attendee.xml",  
    ],  
    "installable": True,  
    "auto_install": False,  
    "application": True,  
}
```

Gambar 68 Panggil `attendee.xml` dari `__openerp__.py`

Susunan file pada addons `academic` kita sejauh ini adalah seperti ini... ada tambahan file `attendee.xml`.

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

```
|-- session.py  
`-- session.xml
```

Lanjut...

Restart server, coba lihat tampilan list view Attendee yang baru...

Daftar Attendee		
		Search...
		Filters ▾ Group By ▾ Favorites ▾
Name	Session	Partner
attende	Session 1	Administrator

Gambar 69 List view Attendee udah lengkap

okee... udah muncul field nya sesuai kemauan kita...

perhatikan kolom Partner dan Session udah langsung muncul nama Partner dan Session yang direlasikan.

9.4 BIKIN RELASI SESSION HAS MANY ATTENDEE

Sesuai ERD sebelumnya, satu **Session** bisa punya banyak **Attendee**. Jadi kita perlu edit definisi class model **Session** supaya mendukung relasi ini.

Edit file `session.py`.

Tambahi field baru namanya `attendee_ids`.

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

Gambar 70 Tambah field attendee_ids di Session clas

Disini kita definisikan kolom baru dengan nama `attendee_ids`. Ingat konvensi sebelumnya, akhiran `_id` artinya dia merupakan kolom relasi dan penambahan huruf “s” sehingga menjadi `_ids` artinya relasi ke banyak record pada table lawan.

Sama seperti Course ke Session, kolom ini punya type `one2many` yang minta parameter sebagai berikut:

Parameter pertama adalah object model lawannya, yaitu yang boleh dimiliki lebih dari satu oleh object ini, disini `academic.attendee`. Artinya satu record di object `session` boleh punya banyak pasangan record di object `attendee`.

Parameter kedua adalah nama field pada object lawan, disini `session_id`. Lihat definisi kolom di object `attendee`, disana ada kolom namanya `session_id` yang menandakan bahwa dia dimiliki oleh suatu record di object `session`.

Parameter ketiga adalah label waktu ditampilkan di view.

Parameter keempat dan seterusnya sifatnya optional, disini kita set `ondelete="cascade"` artinya kalo record pada object `session` dihapus maka record-record terkait pada object `attendee` juga ikut terhapus.

Setelah dideklarasikan seperti diatas, maka object `attendee` udah punya satu kolom tambahan dengan type `one2many` dan udah bisa ditampilkan di view seperti halnya field-field regular lainnya.

9.5 MODIF FORM VIEW SESSION – TAMBAH ATTENDEE

Untuk nampilin kolom tambahan pada object session, kita modif file `session.xml` yang merupakan kumpulan view untuk object ini.

Tambahkan baris ini pada form view Session...

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

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

</sheet>
</form>
</field>
```

Gambar 71 Tambah tab Attendee di Session form view

Disini kita modif form view Session, yaitu nambahin tag `notebook` baru dan satu tag `page` di dalamnya.

Dalam tag `page` kita keluarkan field `attendee_ids` milik object `session`.

Didalam tag `field` kita pasang tag `tree`, dimana disini tag `tree` langsung berkorelasi dengan object yang merupakan relasi dari kolom `attendee_ids` yaitu `attendee`.

Artinya kita bisa munculkan kolom-kolom yang ada pada object `attendee` di dalam tag `tree`. Disini kita munculkan semua kolom yang ada di object `attendee`.

Okee... restart server dan update module.

Lihat hasilnya...

The screenshot shows the 'Session Name' field set to 'Session 2'. Under the 'Course' section, 'PHP' is selected. In the 'Duration' section, there are two dropdown menus: 'Duration' (set to 20) and 'Seats' (set to 20). The 'Start Date' is listed as '02/23/2017'. The 'Active' checkbox is checked. Below this, the 'Attendees' tab is selected, displaying a table with one row: 'Attendee1' (Name) and 'Administrator' (Partner). A link 'Add an item' is visible below the table.

Gambar 72 Tab Attendee muncul di form view Session

Sekarang form Session udah ada tambahan tab baru yang berisi Attendee yang hadir dalam Session tersebut.

Dan kalau di edit, maka kita bisa tambahkan Attendee baru langsung dari form Session, dengan klik link **Add new item**.

The screenshot shows a modal dialog titled 'Create: Attendees'. It has fields for 'Name' (containing 'attendee2') and 'Partner' (containing 'Administrator'). A 'Session' dropdown menu is shown next to the partner field. At the bottom of the dialog are buttons for 'Save & Close', 'Save & New', and 'Discard'.

Gambar 73 Add Attendee dari Session form view

Form Attendee yang muncul disini sama persis dengan ketika kita add Attendee melalui menu Attendee.

Tapi Session boleh dikosongkan disini dan otomatis link dengan Session yang sedang di-edit sekarang.

10 REKAP HARI 1

Wuih..... Udah banyak juga yang kita pelajari di hari pertama ini...

Berikut rekapnya

- mengetahui spesifikasi requirement aplikasi Academic Information System
- desain ERD
- mengetahui struktur Addons odoo
- mulai bikin addons Academic
- bikin Menu dan ActionWindow untuk Course, Session, dan Attendee
- bikin Class Course, Session, dan Attendee
- bikin relasi Course ke Session
- bikin relasi Session ke Attendee
- bikin tree view, form view
- bikin tab di form view

11 HARI 2: INHERITANCE – INSTRUCTOR

Halo broooo.... 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` bertipe Boolean. Nantinya Partner yang bertipe instruktur bisa dipilih di form Session, dan jika tidak maka tidak bisa dipilih di Session sebagai Instruktur.

11.1 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 74 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 75 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
```

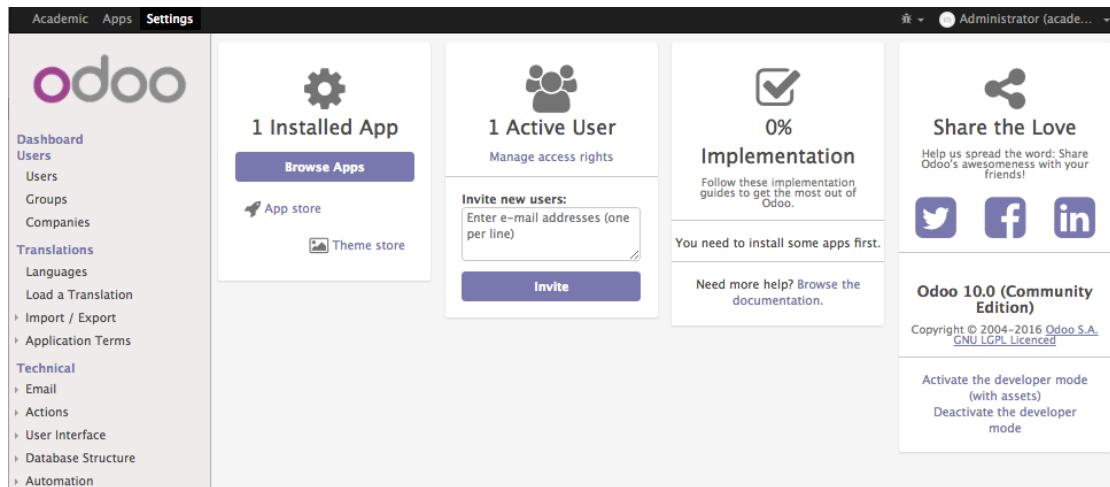
Setelah proses inherit ini, maka field `is_instructor` udah tersedia pada class asli `res.partner`.

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

11.2 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`)...

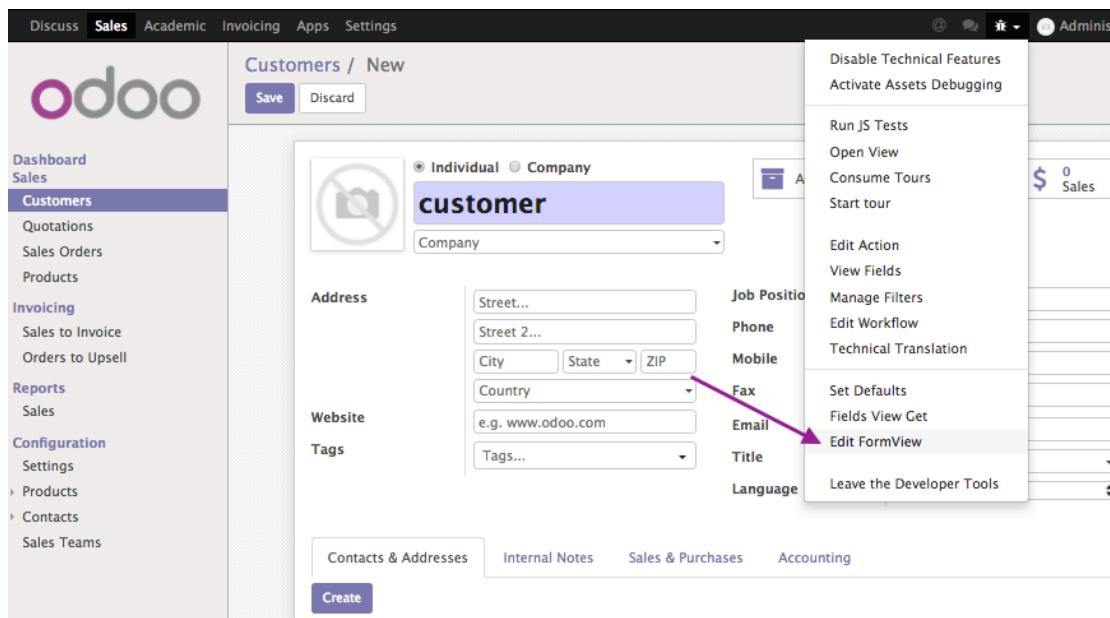


Gambar 76 Activate Developer Mode

Klik Activate the Developer Mode.

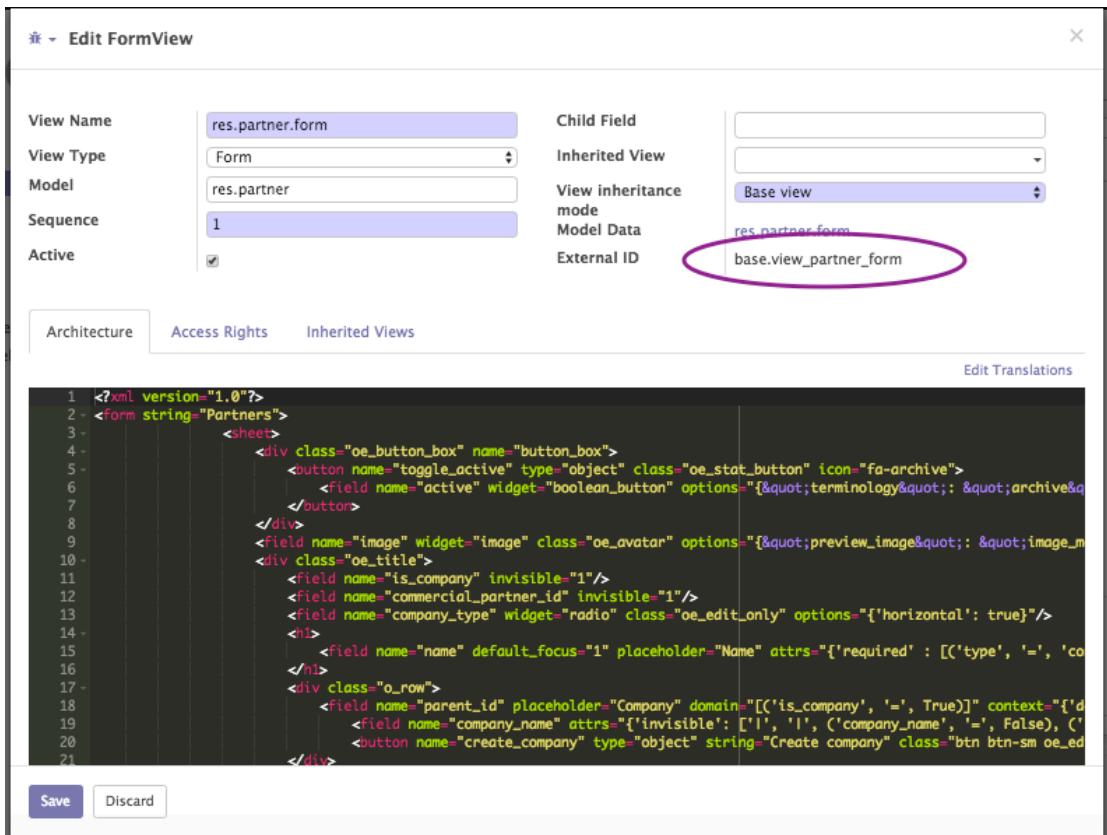
Buka form Partner dari menu **Sale - Customer**. Install module Sale Management jika belum ada menu tersebut.

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



Gambar 77 Lihat definisi form view

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



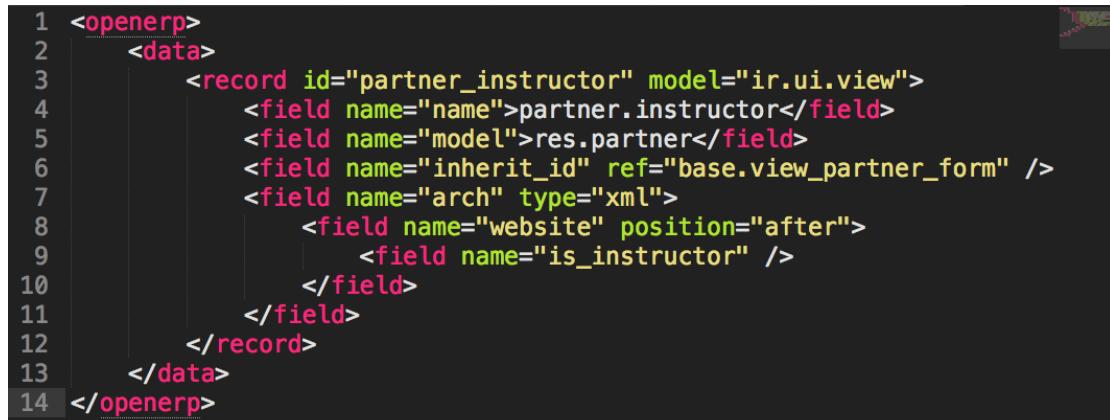
Gambar 78 Definisi form view res.partner

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

Setelah dapet `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 79 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": """\nAcademic Information System Day2

""",
    "data": [
        "menu.xml",
        "course.xml",
        "session.xml",
        "attendee.xml",
        "partner.xml",
    ],
    "installable": True,
    "auto_install": False,
    "application": True,
}
}
```

Gambar 80 Panggil partner.xml dari __openerp__.py

Pada step ini strukur 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:

The screenshot shows the 'Customers / New' screen in Odoo. At the top, there are 'Save' and 'Discard' buttons. Below them is a header with 'Individual' (radio button selected) and 'Company' (radio button). A placeholder 'Name' is shown, with a dropdown menu below it set to 'Company'. To the right of the name are status indicators: 'Active' (green), '0.00 Invoiced' (purple), and '\$ 0 Sales' (blue). The main form area contains fields for 'Address' (Street, Street 2, City, State, ZIP, Country), 'Job Position' (e.g. Sales Director), 'Phone', 'Mobile', 'Fax', 'Email', 'Title', and 'Language' (English). On the left, there are sections for 'Website' (e.g. www.odoo.com), 'Is Instructor' (checkbox checked), and 'Tags' (dropdown menu). Below the main form are tabs: 'Contacts & Addresses' (selected), 'Internal Notes', 'Sales & Purchases', and 'Accounting'. A 'Create' button is at the bottom left.

Gambar 81 Muncul Is Instructor di form Partner

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

11.3 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 82 Tambahi domain di instrutor_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 Duration

Instructor Seats
Agus Anis
Create and Edit...

Start Date Active

Attendees

Name	Partner
Add an item	

Gambar 83 Hanya partner yang `is_instructor` True yang bisa dipilih

11.4 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

Individual Company

Joko

Company

Address Street...
Street 2...
City State ZIP
Country

Job Position e.g. Sal

Website e.g. www.odoo.com

Phone

Mobile

Fax

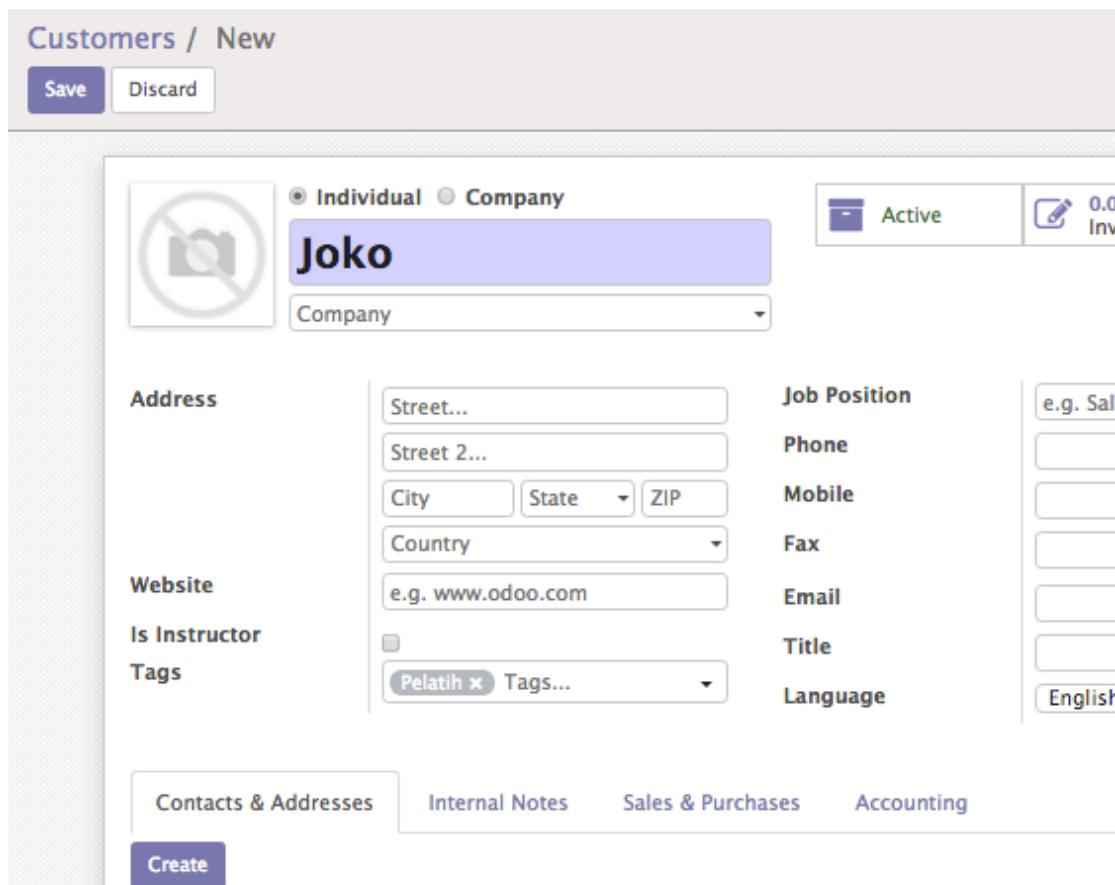
Is Instructor

Email

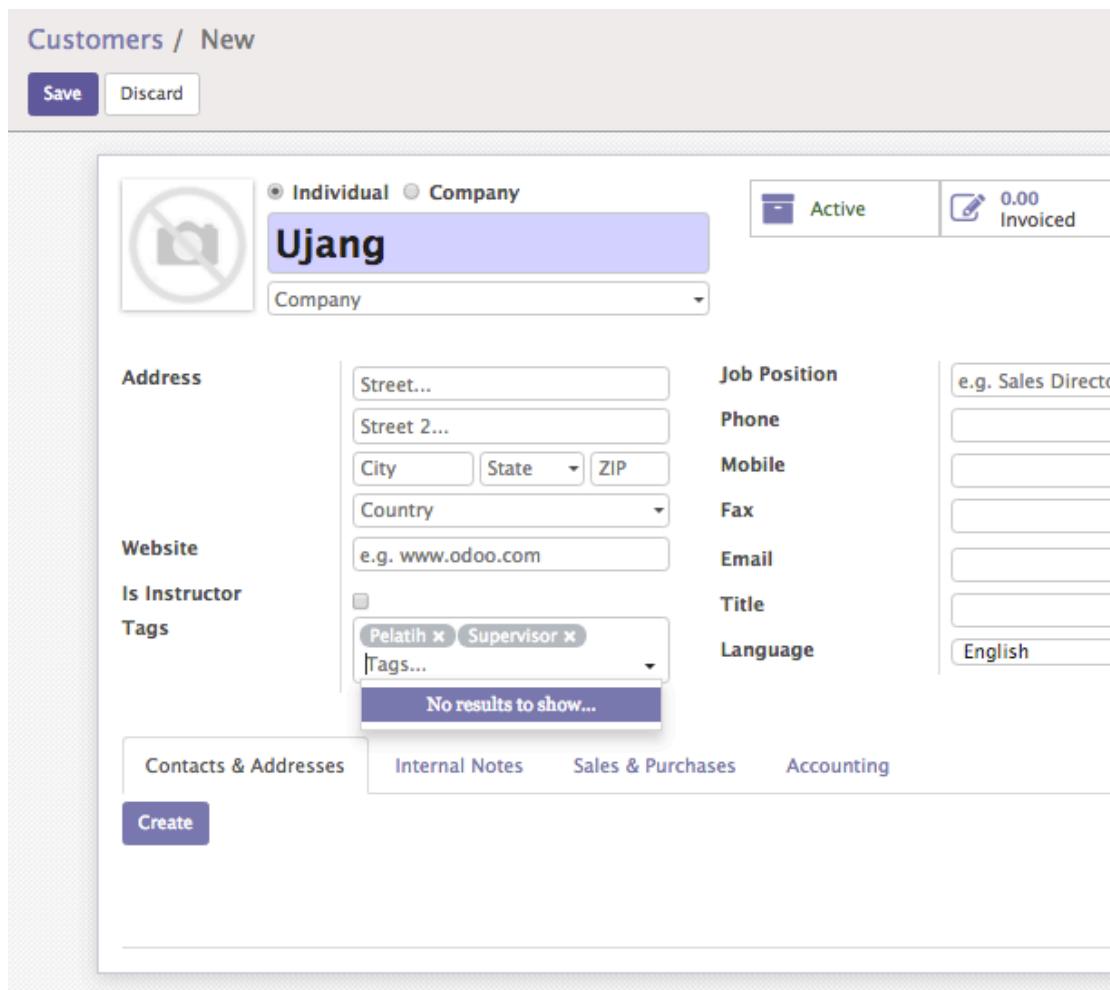
Tags Pelatih Tags...

Title

Language English



Gambar 84 Tambahin kategori partner sebagai Pelatih tapi bukan Instructor



Gambar 85 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="['|',('is_instructor','=',True),
                  ('category_id','=','Pelatih')]/>
    <field name="start_date" />
</group>
```

Gambar 86 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 pemulisannya sama seperti sebelumnya, hanya saja operatornya ditarik 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	<input type="text"/>	Duration	<input type="text" value="0"/>								
Instructor	<input type="text"/>	Seats	<input type="text" value="0"/>								
Start Date	<input type="text"/>	Active	<input type="checkbox"/>								
Attendees	<table border="1"><tr><td>Agus</td><td><input type="checkbox"/></td></tr><tr><td>Anis</td><td><input type="checkbox"/></td></tr><tr><td>Joko</td><td><input type="checkbox"/></td></tr><tr><td>Ujang</td><td><input type="checkbox"/></td></tr></table>			Agus	<input type="checkbox"/>	Anis	<input type="checkbox"/>	Joko	<input type="checkbox"/>	Ujang	<input type="checkbox"/>
Agus	<input type="checkbox"/>										
Anis	<input type="checkbox"/>										
Joko	<input type="checkbox"/>										
Ujang	<input type="checkbox"/>										
Name	Create and Edit...										
Add an item	Partner										

Gambar 87 Partner yang kategori Pelatih bisa dipilih sebagai instruktur

11.5 MENU INSTRUKTUR

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

Pertama, edit file `menu.xml` untuk nambahin menu Instructor. 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 88 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>
            nambah instructor
        </p>
    </field>
</record>
```

Gambar 89 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>
```

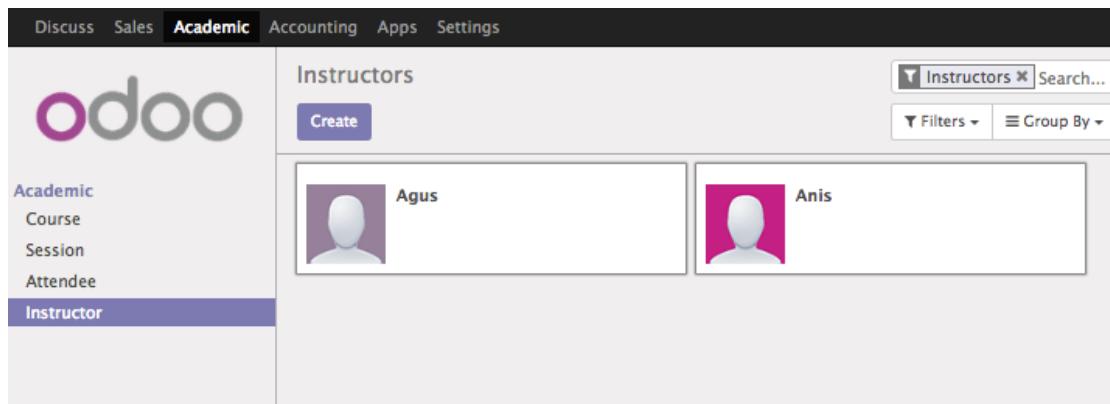
Gambar 90 Default filter partner instructor

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...



The screenshot shows the Odoo Academic interface. The top navigation bar includes links for Discuss, Sales, Academic (which is the active tab), Accounting, Apps, and Settings. On the left, a sidebar lists Academic, Course, Session, Attendee, and Instructor, with Instructor being the selected item. The main content area is titled "Instructors" and contains a "Create" button. Below this, there are two entries: "Agus" and "Anis", each with a small placeholder profile picture. A search bar at the top right is set to "Instructors".

Gambar 91 Menu Instructor muncul dan membuka Partner dengan filter Instructors

12 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.

12.1 DEFINISI FUNCTION FIELDS

Buka file `session.py`, tambahkan satu field baru yang namanya `taken_seats`. Pendefinisianya 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 92 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 93 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 perserta 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`.

12.2 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 94 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>
    </data>
</openerp>

```

Gambar 95 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								
	Create	Import						
	Course	Name	Instructor	Start Date	Duration	Seats	Active	Taken Seat
1		Session1			0	100	<input checked="" type="checkbox"/>	2.00
2								
3								
4								
5								

Gambar 96 Muncul field taken_seats di list view

dan muncul juga pada form view Session.

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

Attendees	
Name	Partner
01	Agus
02	Badu

Gambar 97 Muncul field taken_seats di form view

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

12.3 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 98 Nampilin taken_seats di form dengan progress bar

Restart odoo dan update module, hasilnya...

The screenshot shows the 'Daftar Session / Session1' form. At the top, there are 'Save' and 'Discard' buttons, and a page number '1 / 1'. The form fields include:

- Session Name:** Session1
- Course:** (dropdown menu)
- Instructor:** (dropdown menu)
- Start Date:** (dropdown menu)
- Duration:** 0
- Seats:** 10
- Active:** checked
- Taken Seat:** A progress bar showing 20% completion, with a value of 0 displayed next to it.

Below these fields is a table titled 'Attendees' with columns 'Name' and 'Partner'. It contains two rows of data:

Name	Partner
01	Agus
02	Badu
Add an item	

Gambar 99 Muncul progressbar persentasi seats di form

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

12.4 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.

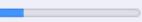
```

<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 100 Nampilin taken_seats di tree dengan progress bar

Restart odoo dan update module, hasilnya...

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

Gambar 101 Muncul progressbar persentasi seats di tree

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

13 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 102 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.

14 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.

14.1 PYTHON CONSTRAINTS

Sebagai contoh kita buat constraints bahwa kalau seorang Partner udan 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 103 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 constraintnya.

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`.

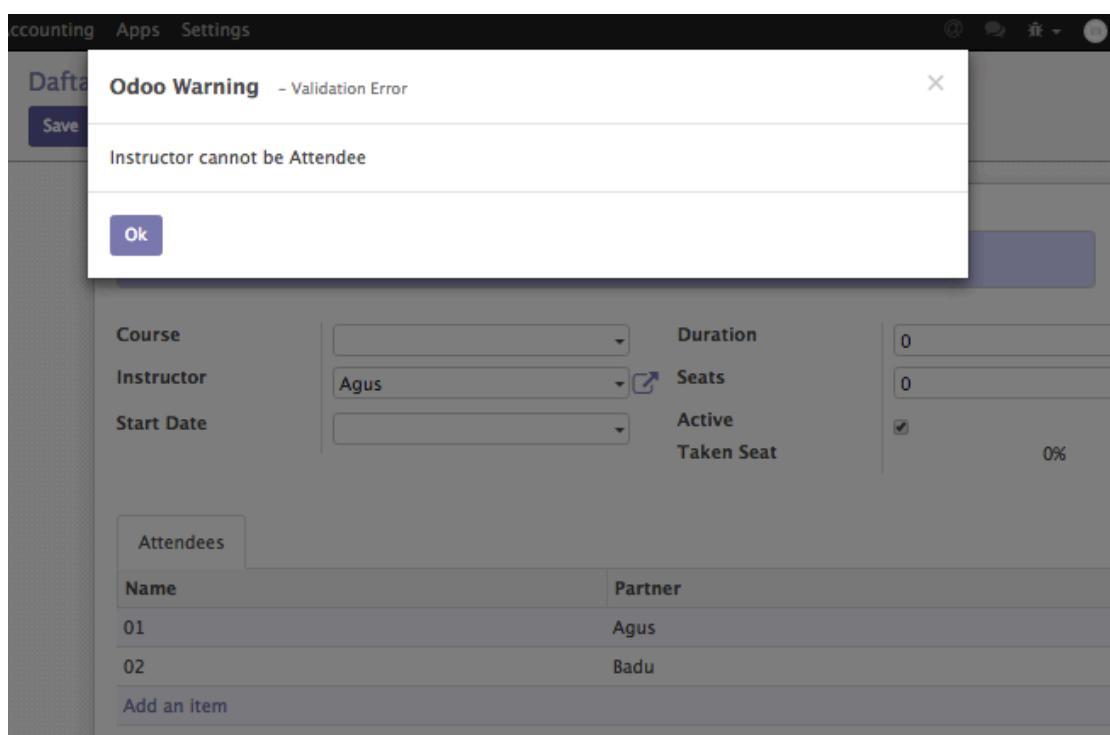
```
if session.instructor_id.id in x:  
    return False
```

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 104 Muncul warning kalau ada data tidak sesuai constraints

14.2 SQL CONTRAINS

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

Namun 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, _

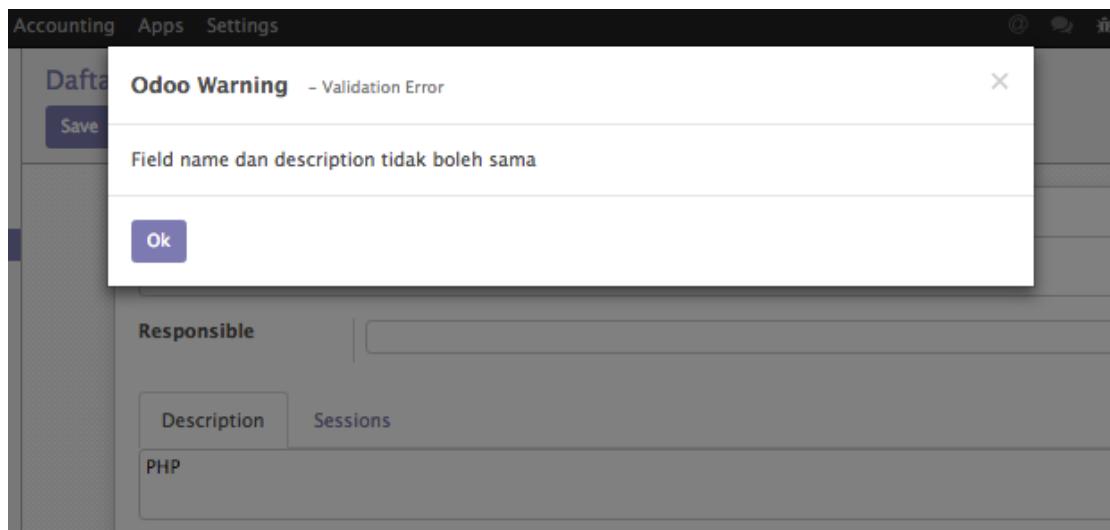
class Course(models.Model):
    _name = 'academic.course'
    _rec_name = 'name'

    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")

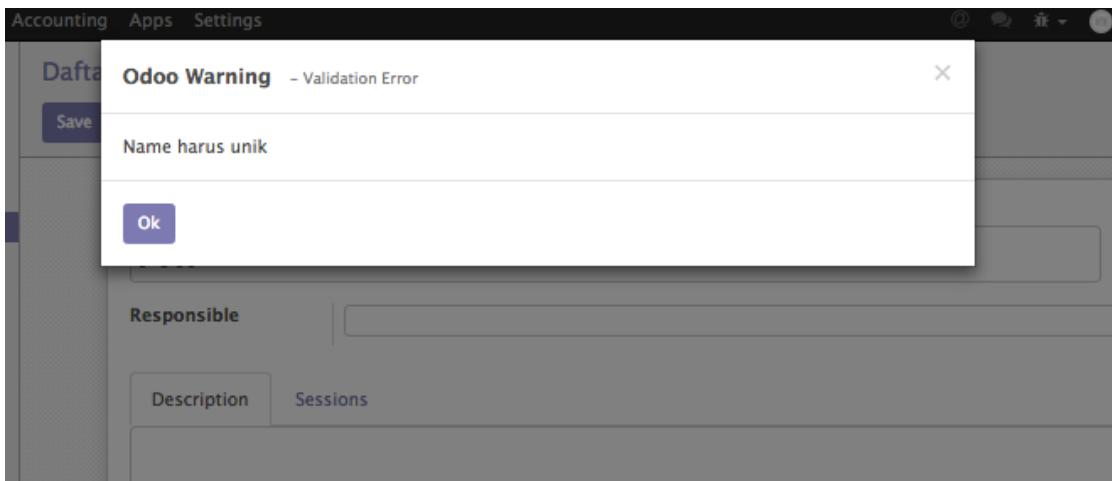
    _sql_constraints = [
        ('cek_name_desc', 'CHECK(name <> description)',
         'Field name dan description tidak boleh sama'),
        ('cek_unik_name', 'UNIQUE(name)',
         'Name harus unik')
    ]
```

Gambar 105 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, _

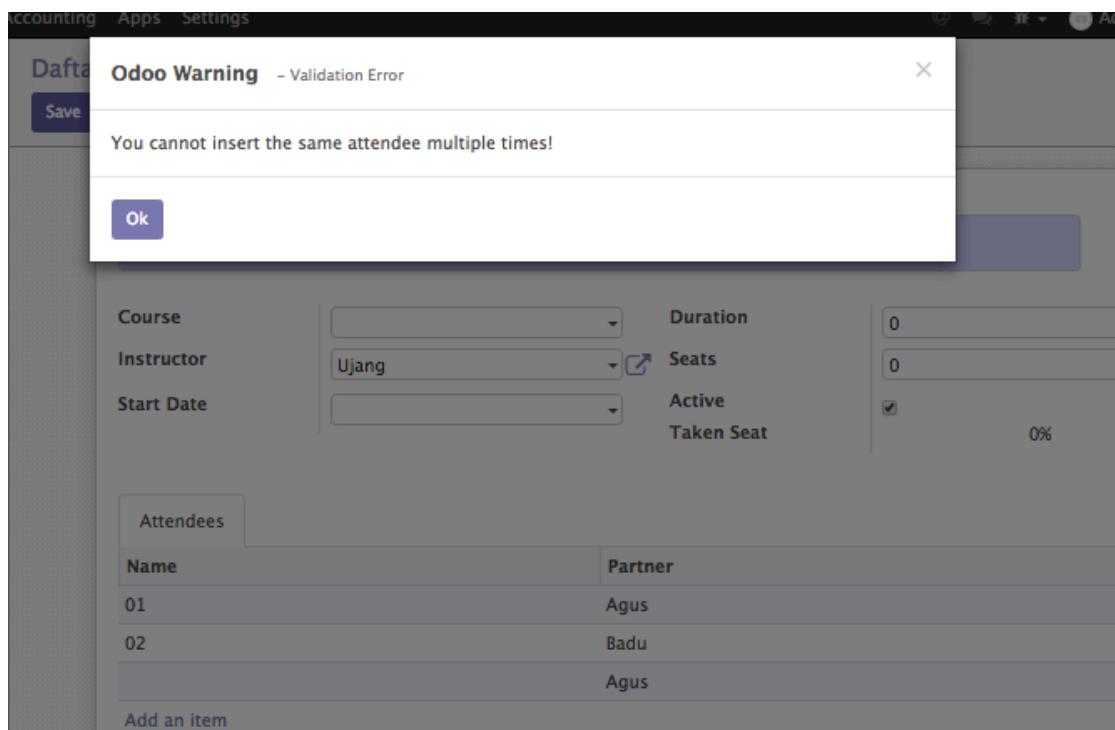
class Attendee(models.Model):
    _name = 'academic.attendee'
    _rec_name = 'name'

    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 106 Cara nambahin SQL constraints di Attendee class

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



15 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 107 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 ...

Daftar Session / New

Save **Discard**

Session Name														
session2														
Course	<input type="text"/>	Duration	<input type="text" value="0"/>											
Instructor	<input type="text"/>	Seats	<input type="text" value="0"/>											
Start Date	<input type="text" value="02/19/2017"/>	Active	<input checked="" type="checkbox"/> Taken Seat 0%											
<table border="1"> <tr> <td>Attendees</td> </tr> <tr> <td>Name</td> <td>Partner</td> </tr> <tr> <td colspan="2">Add an item</td> </tr> <tr> <td colspan="2"></td> </tr> <tr> <td colspan="2"></td> </tr> <tr> <td colspan="2"></td> </tr> </table>				Attendees	Name	Partner	Add an item							
Attendees														
Name	Partner													
Add an item														

Gambar 108 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
```

15.1 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()`.

15.2 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>

16 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 109 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.

Prinspinya 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 ...

The screenshot shows the Odoo interface for managing sessions. At the top, there are buttons for 'Edit' and 'Create', and a dropdown menu labeled 'Action ▾'. A context menu is open over a record named 'session2', with options 'Delete' and 'Duplicate'. The main area displays session details: Course (empty), Instructor (empty), Start Date (02/19/2017), Duration (0), Seats (0), Active (checked), and Taken Seat (0%). Below this, there is a table titled 'Attendees' with columns 'Name' and 'Partner', showing four empty rows.

Gambar 110 Tombol Duplicate

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

Accounting Apps Settings

Administrator (aca)

Daftar Session / Copy of session2

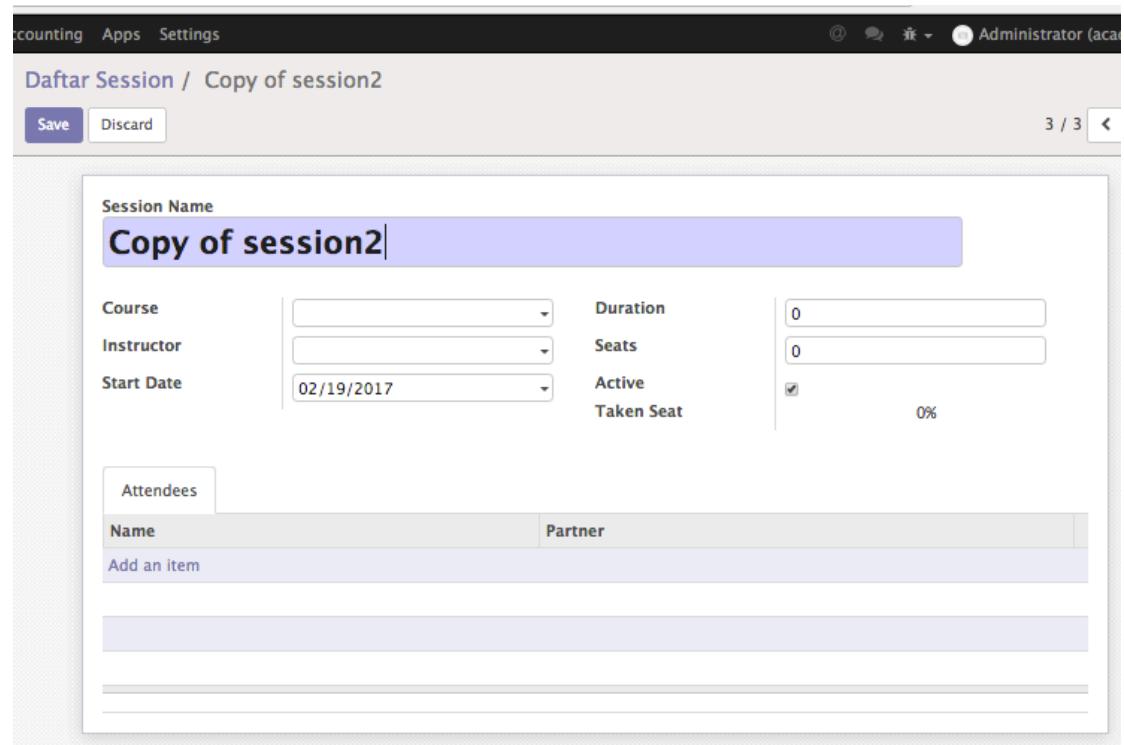
Save Discard 3 / 3 <

Session Name
Copy of session2

Course	Duration
Instructor	Seats
Start Date	Active Taken Seat

Attendees

Name	Partner
Add an item	



Gambar 111 Nilai record yang diduplicate sudah berubah

17 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()`.

18 HARI 3: ADVANCED VIEW

18.1 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>50; green:taken_seats<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 112 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>50; green:taken_seats<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	<input checked="" type="checkbox"/>	(7 records)	<div style="width: 100%;"><div style="width: 100%;"> </div></div>
	Session 2	Java	Badu	09/04/2014	0	3	<input checked="" type="checkbox"/>	(2 records)	<div style="width: 100%;"><div style="width: 33%;"> </div></div>
	Session2	Java	Agus	09/02/2014	0	0	<input checked="" type="checkbox"/>	(2 records)	<div style="width: 100%;"><div style="width: 0%;"> </div></div>
	Session 4	Java	Ujang	09/02/2014	0	0	<input checked="" type="checkbox"/>	(0 records)	<div style="width: 100%;"><div style="width: 0%;"> </div></div>
	Copy of Session 1	Java	Joko	09/02/2014	0	5	<input checked="" type="checkbox"/>	(3 records)	<div style="width: 100%;"><div style="width: 100%;"> </div></div>
	Copy of Copy of Session 1	Java	Joko	09/02/2014	0	20	<input checked="" type="checkbox"/>	(3 records)	<div style="width: 100%;"><div style="width: 100%;"> </div></div>
	Copy of Session 4	Java	Ujang	09/02/2014	0	0	<input checked="" type="checkbox"/>	(0 records)	<div style="width: 100%;"><div style="width: 0%;"> </div></div>
	Session 1	PHP	Badu	09/03/2014	0	0	<input checked="" type="checkbox"/>	(0 records)	<div style="width: 100%;"><div style="width: 0%;"> </div></div>

Gambar 113 Warna tree view udah berubah sesuai kondisi

18.2 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 114 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 115 Aktivasi icon calendar view

Restart odoo dan update module, hasilnya...

The screenshot shows the Odoo interface for managing sessions. At the top, there's a search bar and navigation buttons for Today, Day, Week, and Month. The main area is titled "Daftar Session (Week 8)" and shows a weekly calendar grid from February 19 to 25, 2017. The grid has columns for each day of the week and rows for time intervals from 00:00 to 08:00. Several sessions are listed in the grid, such as "session1" on Monday, "session2" on Wednesday, and "session3" on Thursday. To the right of the calendar, there's a sidebar with a monthly calendar for February 2017, where the 21st is marked as the current date. Below the monthly calendar, there's a checkbox for "php".

Gambar 116 Session ditampilkan dalam calendar

18.3 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 berdasarkan 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 117 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...

Course	Name	Instructor	Start Date	0	100
php	session1		02/20/2017	0	100 <input checked="" type="checkbox"/>
php	session2		02/22/2017	0	0 <input checked="" type="checkbox"/>
php	session3		02/23/2017	0	0 <input checked="" type="checkbox"/>

Gambar 118 Search dengan nama session

Bisa search berdasarkan nama instruktur...

Daftar Session

Course	Name	Instructor	Start Date	Duration	Seats	Active
php	session1		02/20/2017	0	100	<input checked="" type="checkbox"/>
php	session2		02/22/2017	0	0	<input checked="" type="checkbox"/>
php	session3		02/23/2017	0	0	<input checked="" type="checkbox"/>

Gambar 119 Search dengan nama Instruktur

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

Daftar Session

Course	Name	Instructor	Start Date	Duration	Seats	Active
php	session1		02/20/2017	0	100	<input checked="" type="checkbox"/>
php	session2		02/22/2017	0	0	<input checked="" type="checkbox"/>
php	session3		02/23/2017	0	0	<input checked="" type="checkbox"/>

Gambar 120 Filter yang durasinya nggak nol

Bisa kelompokin Session berdasarkan Course-nya...

Daftar Session

Course	Name	Instructor	Start Date
php (3)			

Gambar 121 Group berdasarkan Course

Dan bisa kelompokin session berdasarkan Start Date nya

Daftar Session					Date	Search...
	Course	Name	Instructor	Start Date	Course	Act
February 2017 (3)						
	php	session1		02/20/2017	0	0
	php	session2		02/22/2017	0	0
	php	session3		02/23/2017	0	0

Gambar 122 Group berdasarkan tanggal

18.4 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 123 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 124 Definisi gantt view

Sama seperti record yang lainnya, view ini punya field

`name=session.gant`, dan field

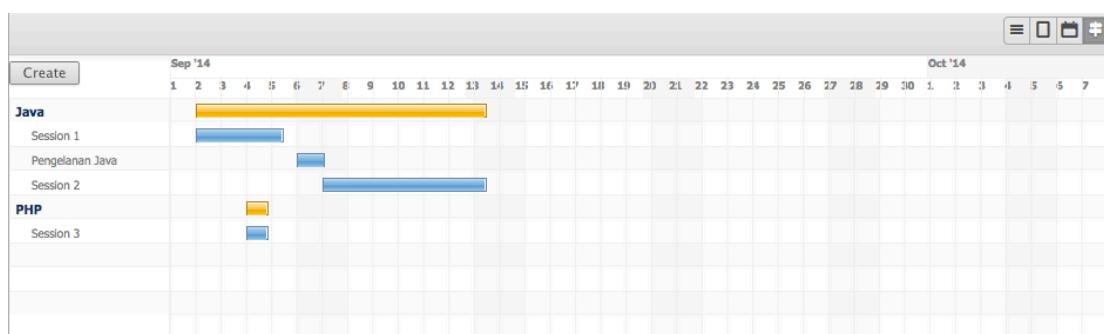
`model=academic.session`.

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 125 Session dalam gantt view per Course

18.5 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 126 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 127 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 grafik 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.

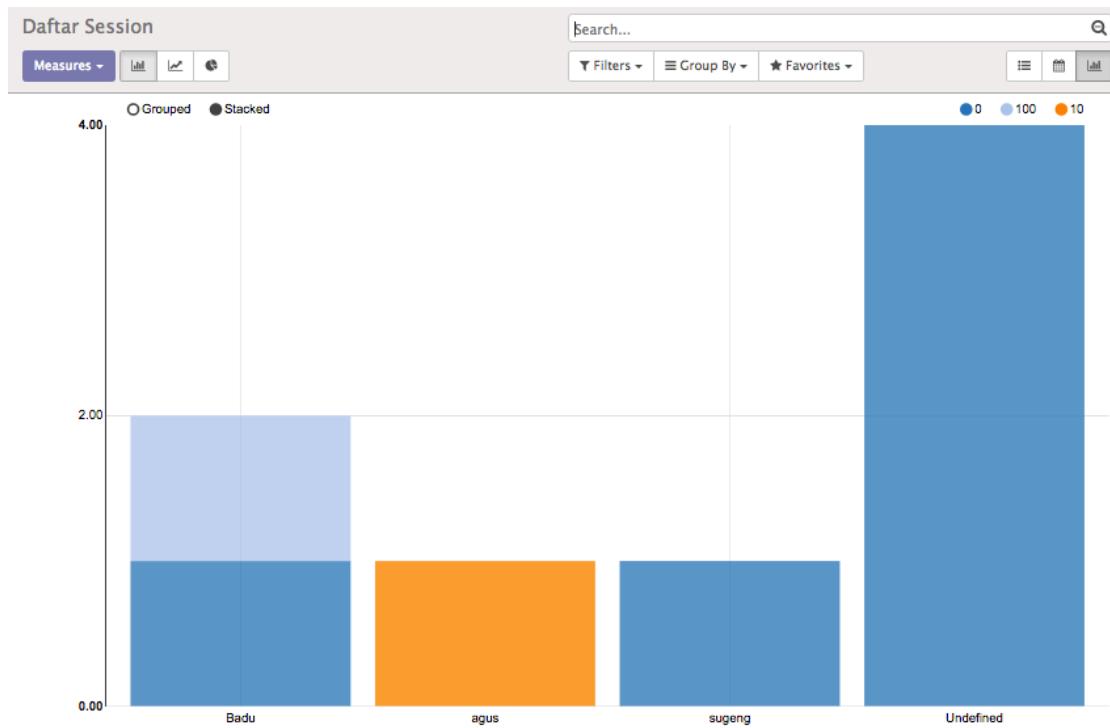
Field pertama akan menjadi data sumbu X, field kedua menjadi sumbu Y, dan field ketiga jika ada menjadi sumbu Z.

Field kedua dan ketiga boleh ada tambahan atribut:

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

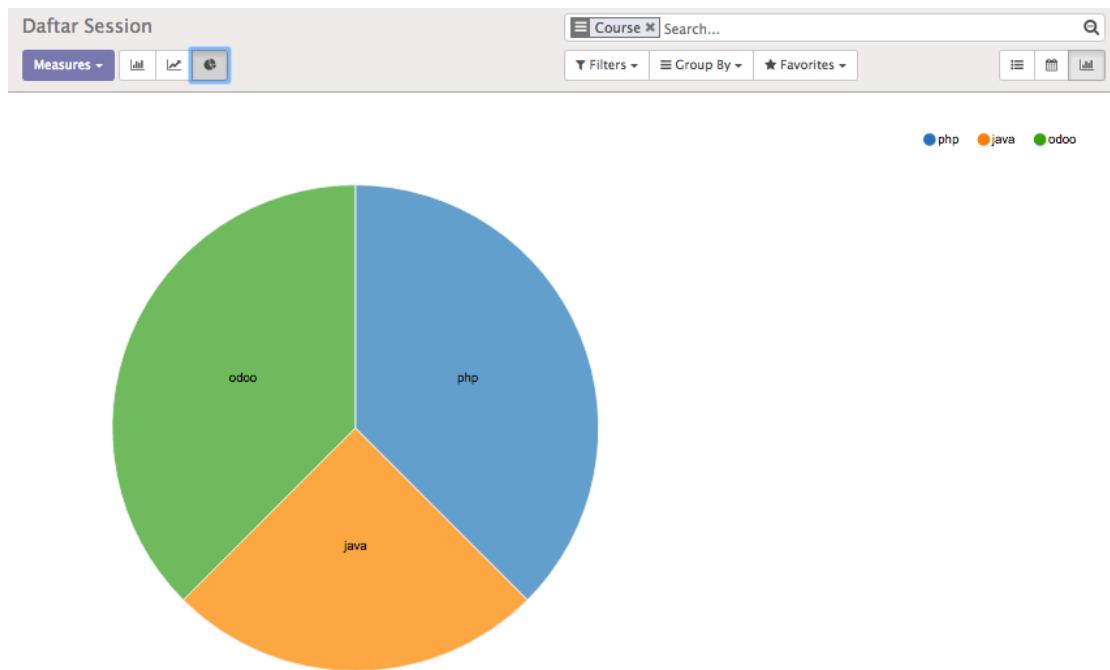
Restart odoo dan update module.

Hasilnya...



Gambar 128 Graph berapa jumlah total seat per instruktur dalam bar chart

Pilih group by Course dan type pie chart...



Gambar 129 Graph berapa jumlah total seat per course dalam pie chart

18.6 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 130 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">
                            
                        </a>
                    <div class="oe_kanban_details">
                        <h4>
                            <a type="open"><field name="name"/></a>
                        </h4>
                        <ul>
                            <li>
                                Seats: <field name="seats"/>
                            </li>
                            <li>
                                Taken Seats:
                                <field name="taken_seats"/>
                            </li>
                        </ul>
                    </div>
                </t>
            </templates>
        </kanban>
    </field>
</record>
```

```
</div>
</t>
</templates>
</kanban>
</field>
</record>
```

Gambar 131 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...

Daftar Session		
<input type="text" value="Search..."/> Filters Group By 1-8 / 8 Favorites		
 session1 Seats: 100 Taken Seats: 3.00	 session2 Seats: 0 Taken Seats: 0.00	 session3 Seats: 0 Taken Seats: 0.00
 odoo1 Seats: 10 Taken Seats: 0.00	 odoo2 Seats: 0 Taken Seats: 0.00	 odoo3 Seats: 0 Taken Seats: 0.00
 java1 Seats: 0 Taken Seats: 0.00	 java2 Seats: 0 Taken Seats: 0.00	

Gambar 132 Session dalam kanban view

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

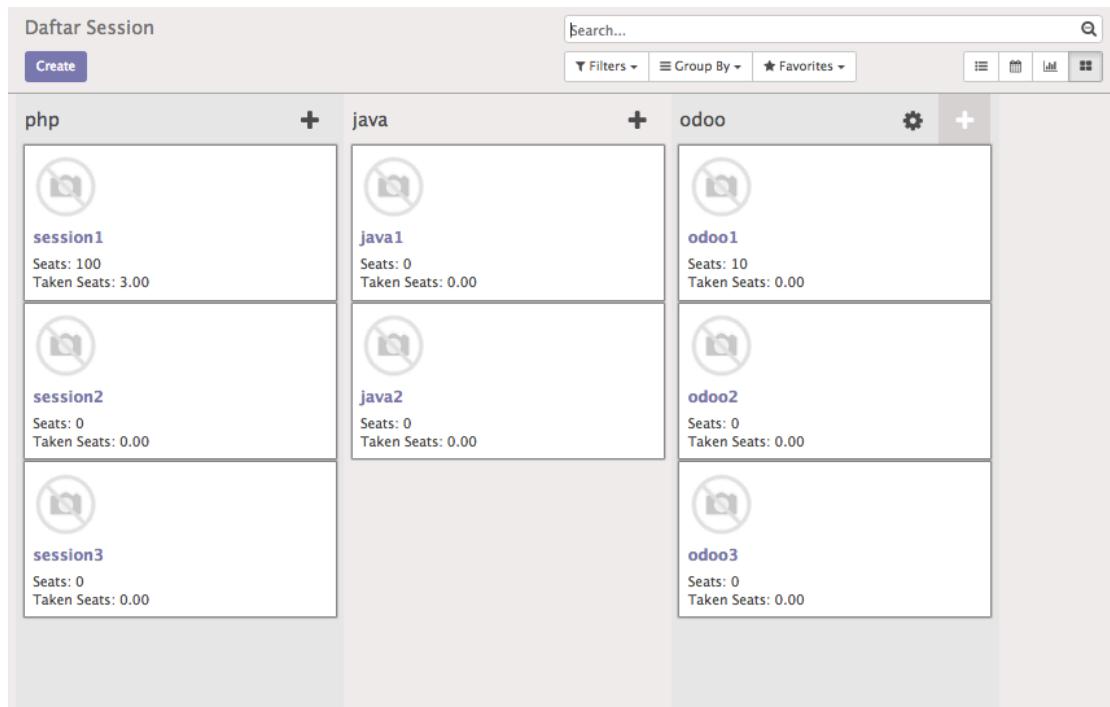
Contohnya kita mau kelompokin berdasarkan Course. Tambahi atribut `default_group_by="course_id"` pada tag kanban.

```
<!-- 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 133 Session di-group per Course

Restart odoo dan update module.

Hasilnya...



Gambar 134 Kanban session per Course

18.7 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 135 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.

```

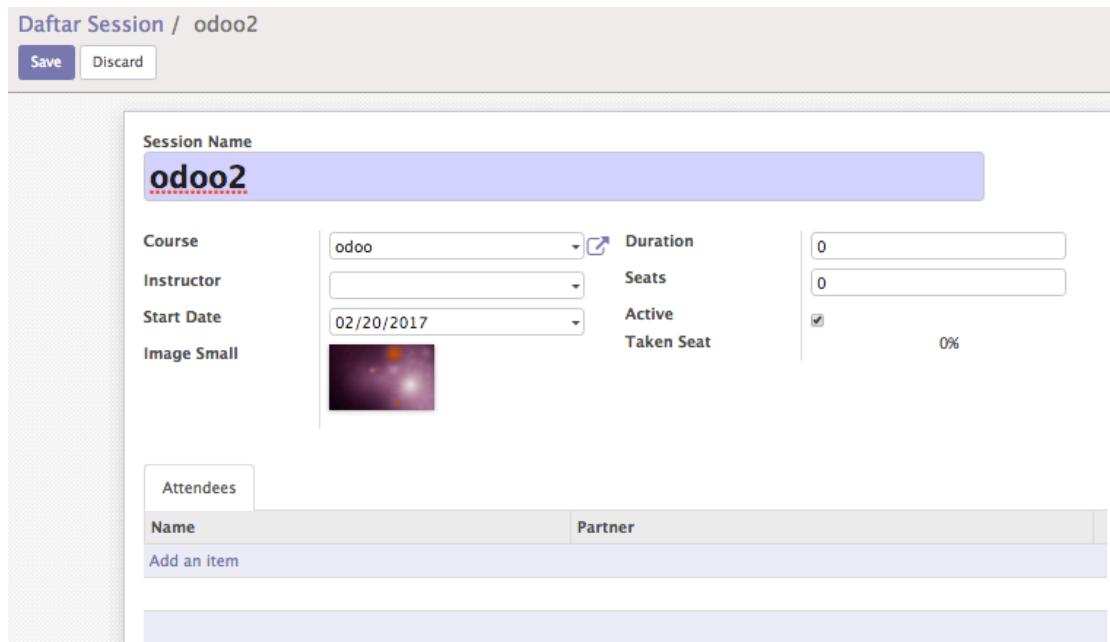
<record id="view_academic_session_form" model="ir.ui.view">
    <field name="name">academic.session.form</field>
    <field name="model">academic.session</field>
    <field name="type">form</field>
    <field name="priority" eval="8"/>
    <field name="arch" type="xml">
        <form string="Session">
            <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 136 Munculkan field `image_small` di form view Session

Restart odoo dan update module.

Hasilnya...



Gambar 137 Field image_small muncul sebagai image field, bisa upload image

Cek lagi kode template kanban view pada file `session.xml`, tag `img` attribut `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">
                            
                        </a>
                    </div>
                </t>
            </templates>
        </kanban>
    </field>
</record>
```

Gambar 138 Munculkan image_small pada kanban view

Restart odoo dan update module.

Hasil di kanban Session...

Daftar Session		
Search...	Filters ▾	Group By ▾
Create	★ Favorites ▾	☰
php	+	java
 session1 Seats: 100 Taken Seats: 3.00		 java1 Seats: 0 Taken Seats: 0.00
 session2 Seats: 0 Taken Seats: 0.00		 java2 Seats: 0 Taken Seats: 0.00
 session3 Seats: 0 Taken Seats: 0.00		 odo01 Seats: 10 Taken Seats: 0.00
		 odo02 Seats: 0 Taken Seats: 0.00
		 odo03 Seats: 0 Taken Seats: 0.00

Gambar 139 Image muncul pada kanban

18.8 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 140 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="manyZone"`, 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 141 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...

	Name	Session	Course
✓ php (3)	01	session1	Course
	02	session1	Session
	03	session1	Partner

Gambar 142 Attendee bisa di-group berdasarkan Course

19 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.

19.1 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 143 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 144 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 145 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 146 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**.

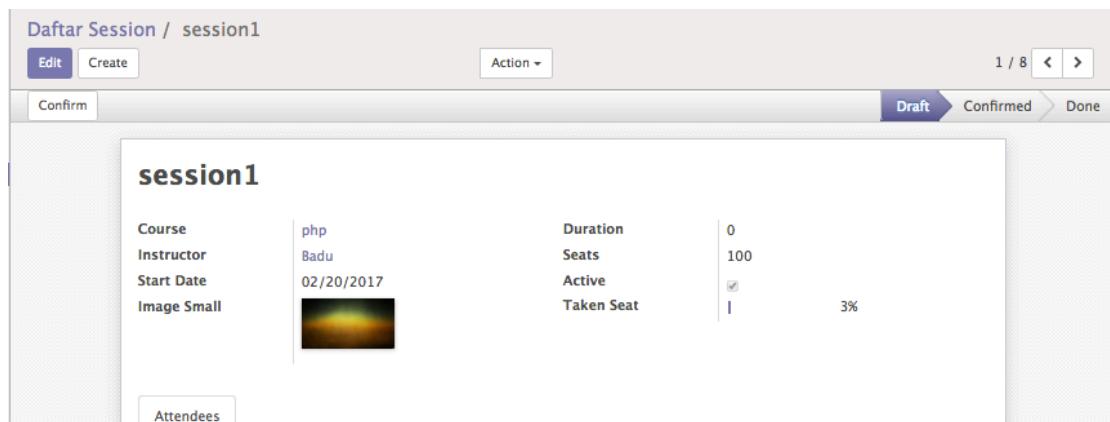
Tombol **Mark as Done** akan memanggil method **action_done**, dan muncul hanya ketika state sudah **Confirmed**.

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 147 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.... 😞

19.2 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**

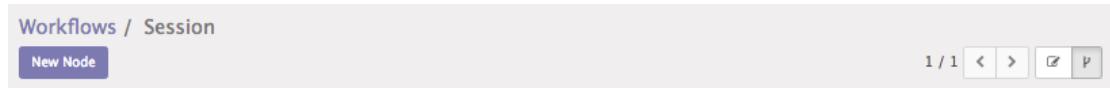
dan **Create** workflow baru.



Gambar 148 Membuat diagram workflow

Masuk ke diagram view dan tambahi nodes dan transitions.

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



Gambar 149 Diagram view

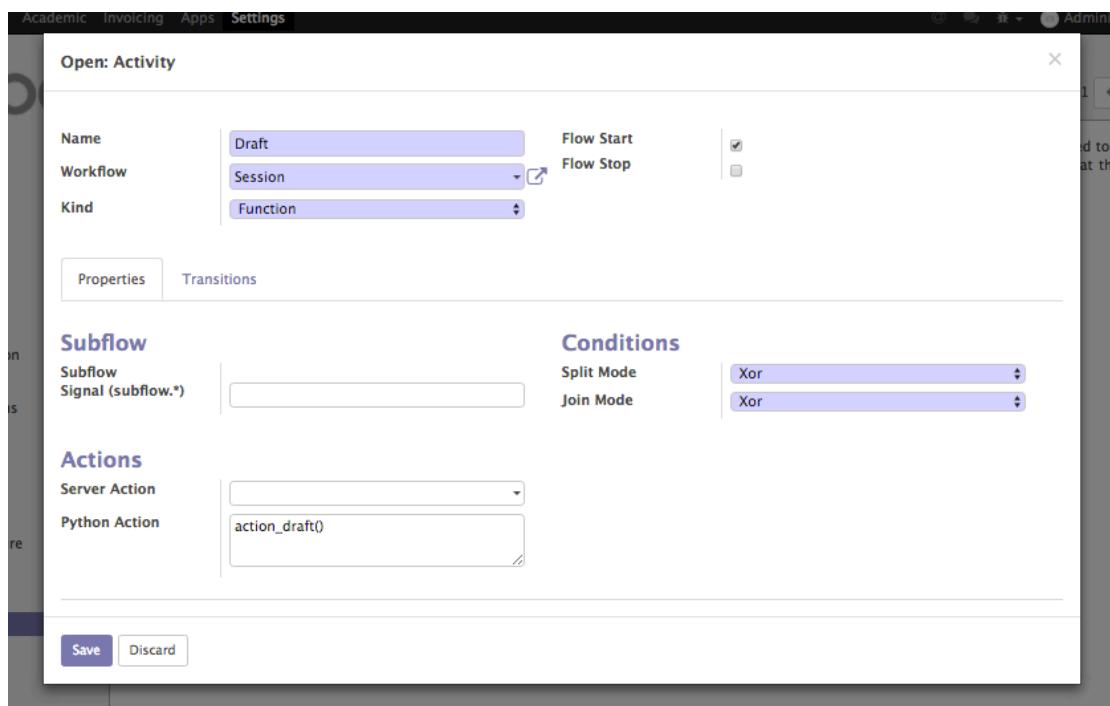
Klik **New Node**:

Name = Draft

Kind = Function

Python Action = action_draft()

Klik **Save**.



Gambar 150 Tambah node activity Draft

Klik **New Node** lagi:

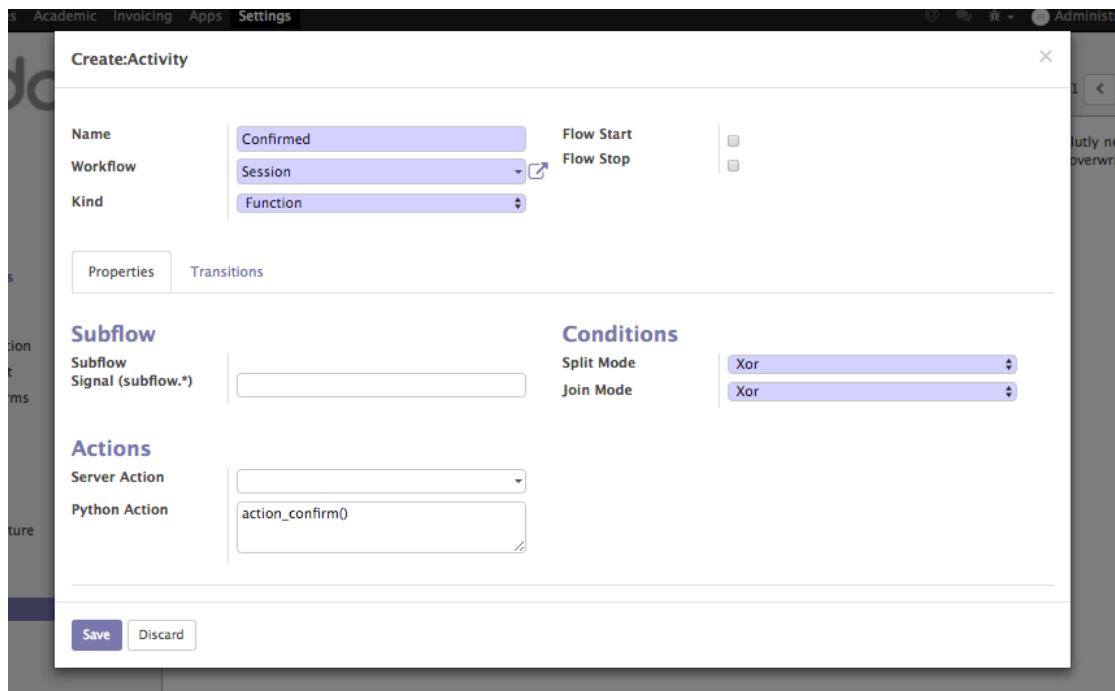
Name = Confirmed

Kind = Function

Python Action = action_confirm()

Flow Start = ya

Klik **Save**.



Gambar 151 Tambah node activity Confirmed

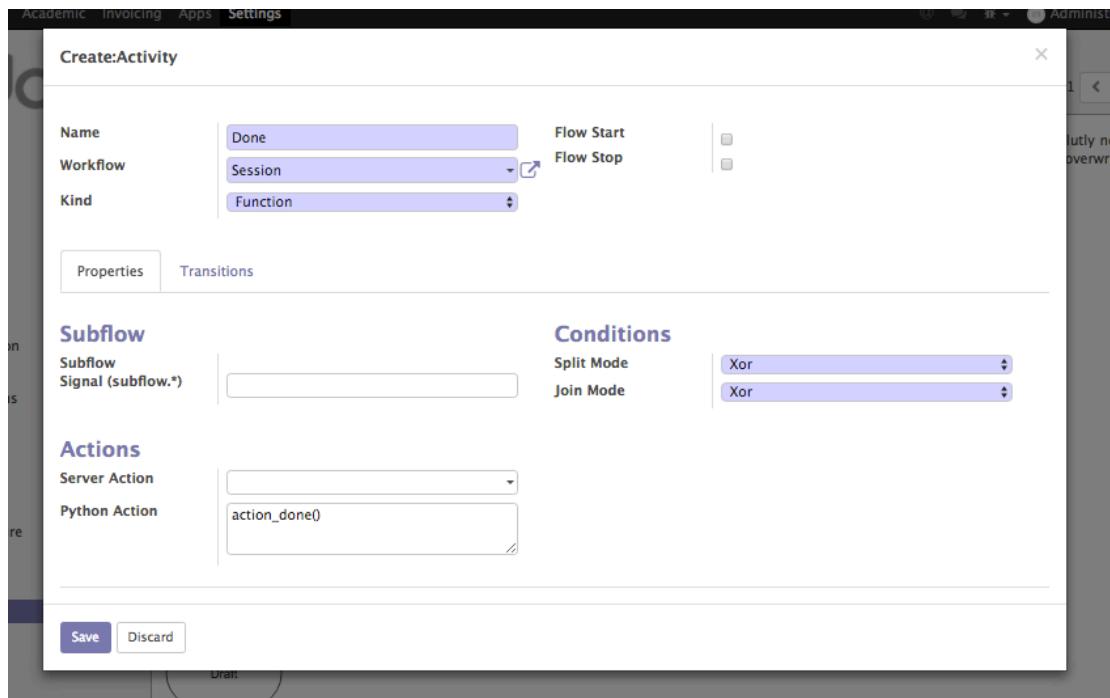
Klik **New Node** lagi:

Name = Done

Kind = Function

Python Action = action_done ()

Klik **Save**.



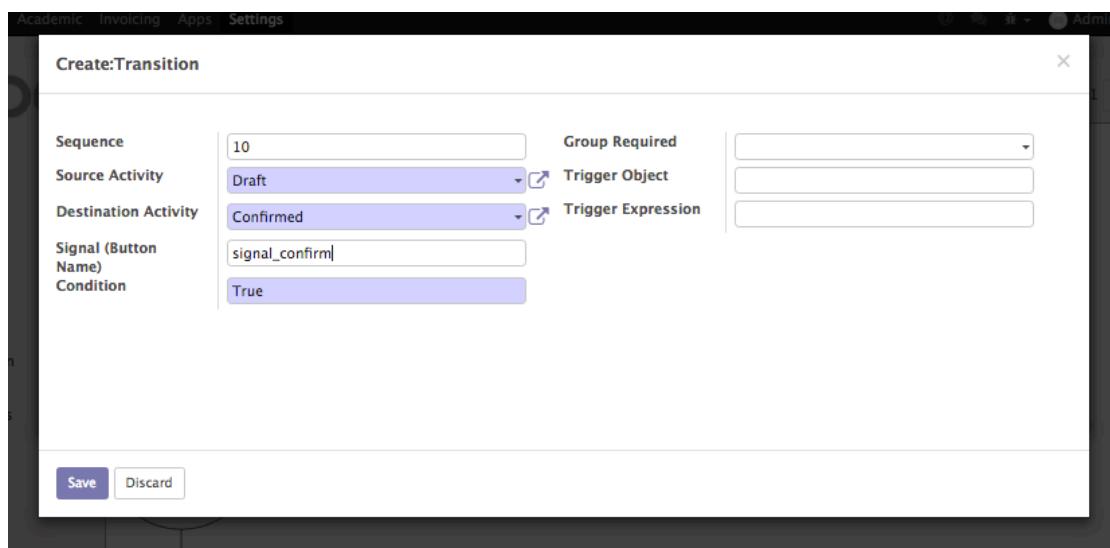
Gambar 152 Tambah node activity Done

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**.

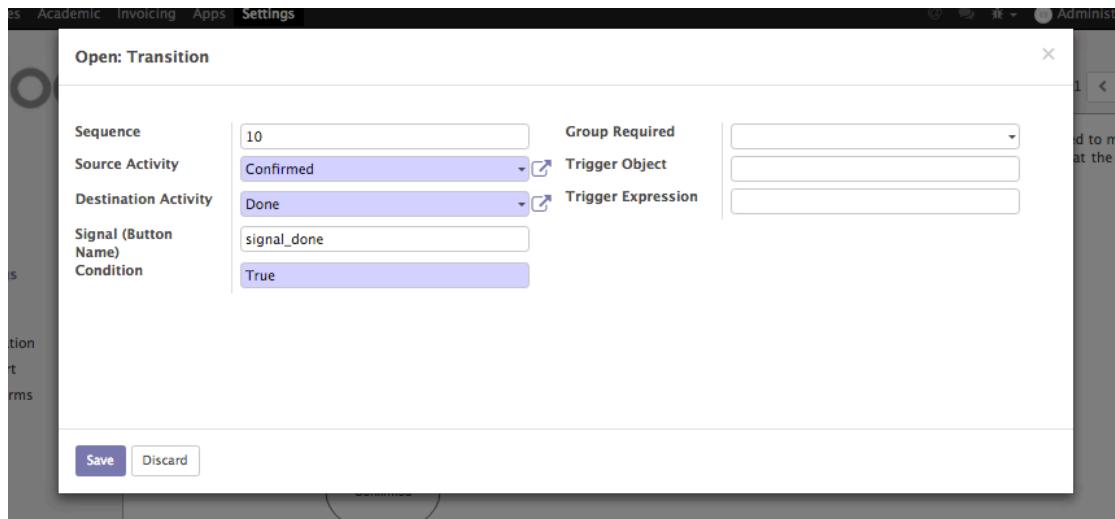
Signal Name = signal_confirm



Gambar 153 Tambah signal confirm

Panah dari **Confirmed** → **Done**.

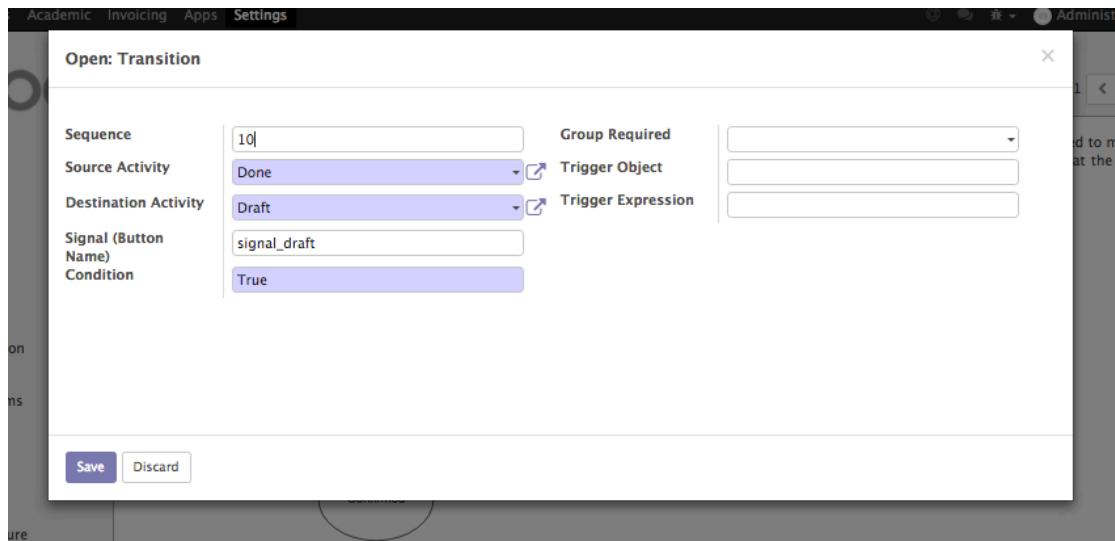
Signal Name = signal_done



Gambar 154 Tambah signal done

Panah dari **Done** → **Draft**.

Signal Name = signal_draft



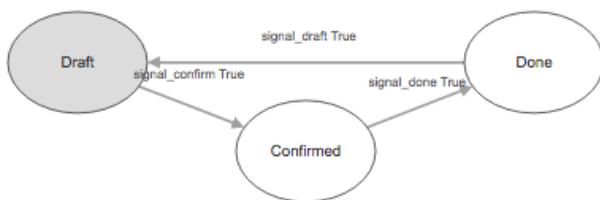
Gambar 155 Tambah signal draft

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

Workflows / Session

New Node

When customizing a workflow, be sure you do not modify an existing node or arrow, but rather, you can only change fields that are empty or set to the default value. If you do upgrade to a future version of Odoo.



Gambar 156 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" />
            </header>
            <field name="state" widget="statusbar" />
        </form>
    </field>
</record>
```

Gambar 157 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.

19.3 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

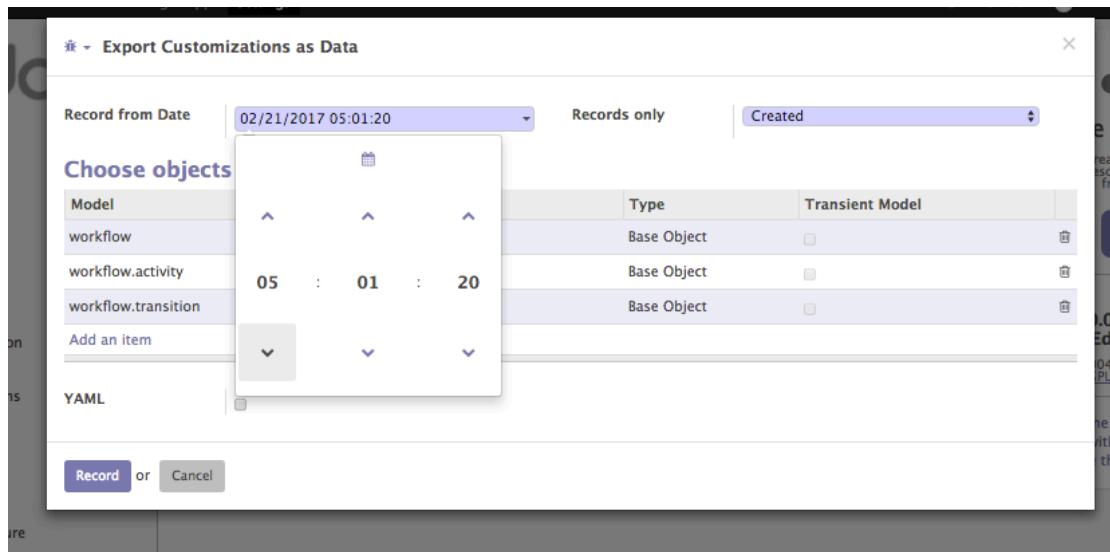
https://www.odoo.com/apps/modules/10.0/base_module_record/

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.

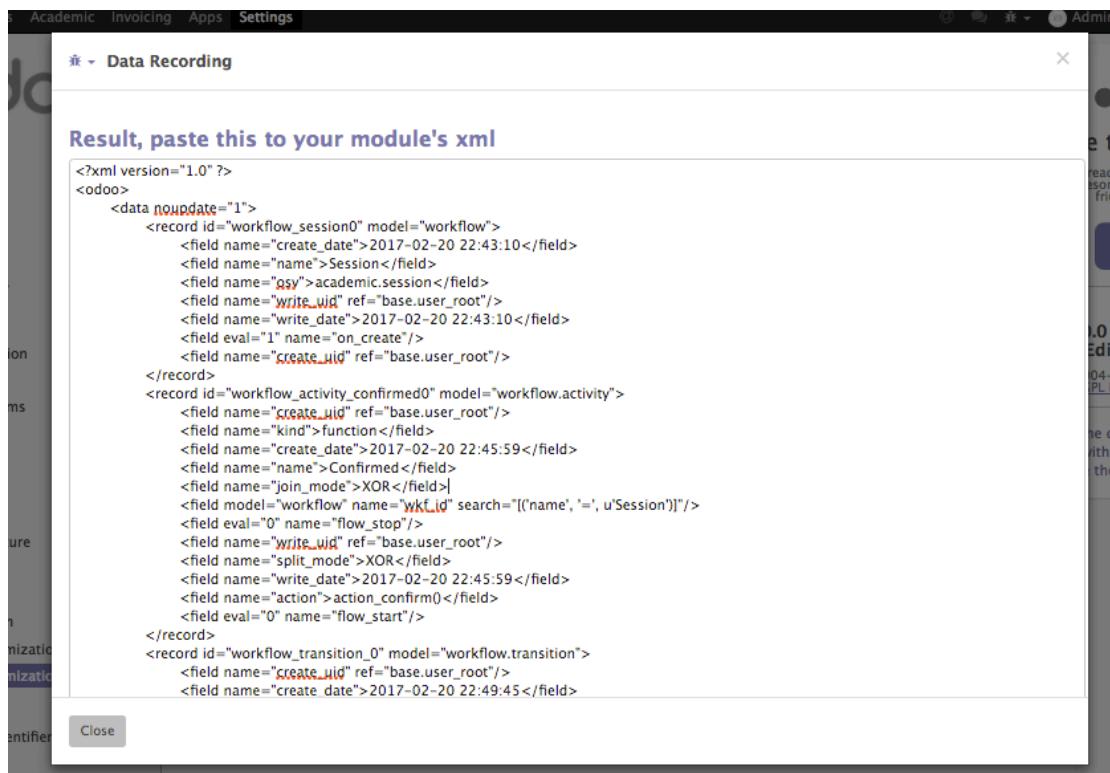
Pilih object `workflow`, `workflow activity` dan `workflow transition`.

Klik `Record`.



Gambar 158 Export workflow ke XML

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



Gambar 159 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>
    </data>
</openerp>
```

```

<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 160 Panggil workflow.xml dari __openerp__.py

20 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

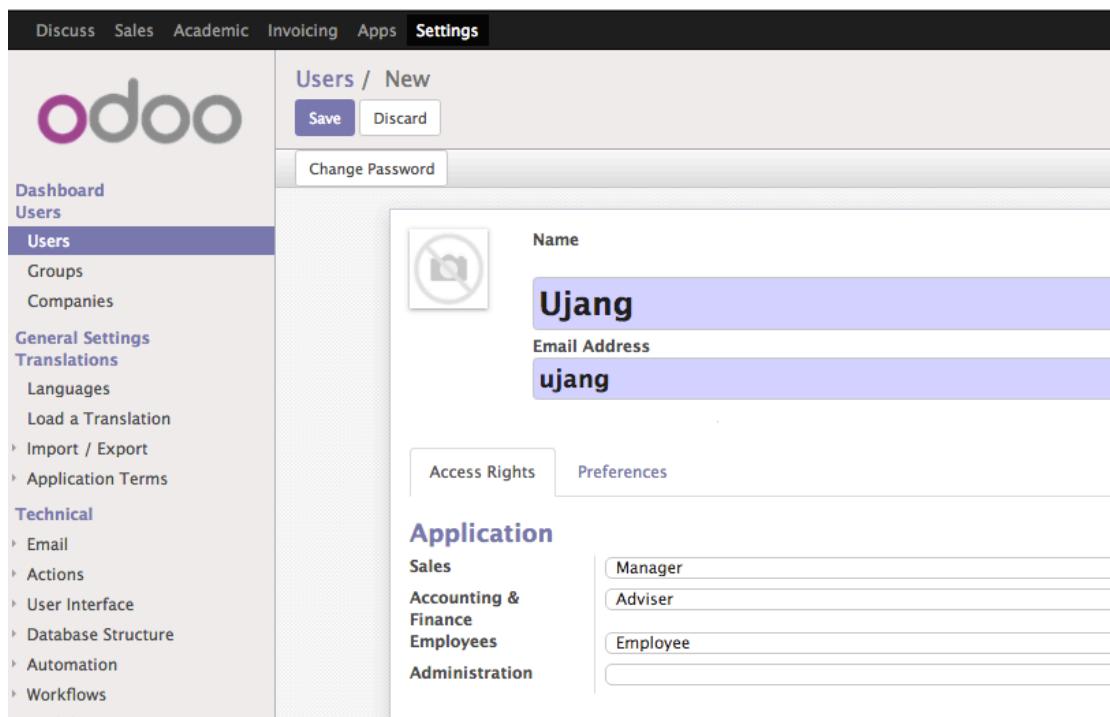
21 HARI 4: SECURITY

odoo udah nyediain sytem security yang cukup bagus, dimana kita bisa batasi akses menu terhadap group user, akses read, create, update, dan delete terhadap object, dan pembatasan akses terhadap record sesuai kriteria tertentu.

21.1 BIKIN GROUP LEWAT INTERFACE

Kita coba culu bikin kontrol akses group melalui user interface.

Melalui menu **Settings > Users > User**, create user baru “Ujang”.



Lalu create group baru yang hanya bisa read Session, Attendee, dan Course, dengan nama “Session Read” melalui menu **Settings > Users > Groups**.

Groups / New

Save Discard

Application	Name
	Session Read

Portal Share Group

Object	Read Access	Write Access	Create Access	Delete Access	Name
academic.course	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	course
academic.attendee	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	attendee
academic.session	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	session

Add an item

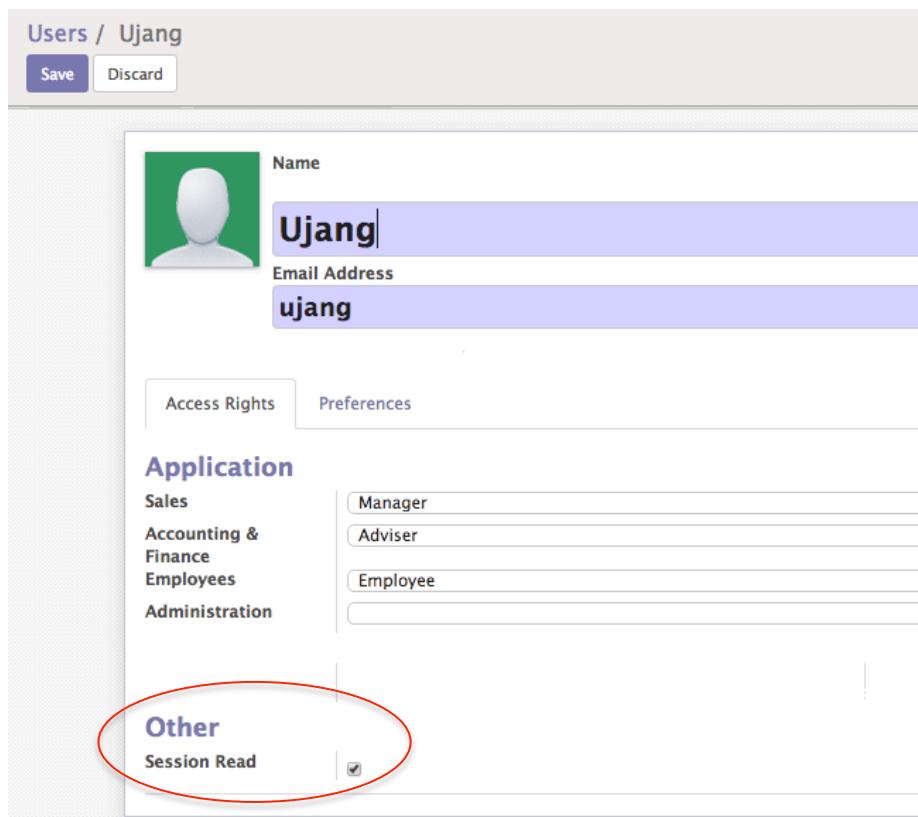
Gambar 162 Access Right group terhadap model

Kolom Name diisi dengan nama Group.

Pada tab Access Right, masukkan object session, attendee, dan course dengan Read Access.

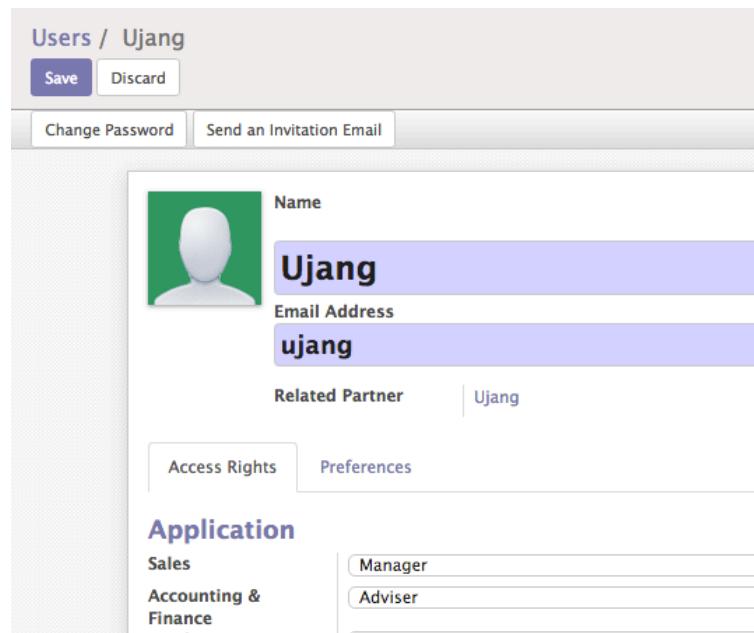
Kolom name hanya informasi saja tapi harus diisi sesuai nama aksesnya misalnya session-read.

Edit user “Ujang” dan masukkan ke group “Session Read”.



Gambar 163 Set User ke Group

Kasi password "Ujang" melalui tombol More > Change Password supaya dia bisa login.



Gambar 164 Change User password

Oke sekarang bisa log in sebagai "Ujang" dan cek bagaimana akses ke Session.

Course	Name	Instructor	Start Date	Duration	Seats	Active	Taken Seat
php	session1	Badu	02/20/2017	0	100	<input checked="" type="checkbox"/>	
php	session2	sugeng	02/22/2017	0	0	<input type="checkbox"/>	
php	session3	Badu	02/23/2017	0	0	<input checked="" type="checkbox"/>	
odoo	odoo1	agus	02/20/2017	0	10	<input checked="" type="checkbox"/>	
odoo	odoo2		02/20/2017	0	0	<input checked="" type="checkbox"/>	
odoo	odoo3		02/20/2017	0	0	<input checked="" type="checkbox"/>	
java	java1		02/20/2017	0	0	<input checked="" type="checkbox"/>	
java	java2		02/20/2017	0	0	<input checked="" type="checkbox"/>	
odoo	sesion3	agus	02/20/2017	0	0	<input checked="" type="checkbox"/>	

Gambar 165 Access user terhadap object sudah terbatas

Oke,... dia bisa lihat (read) Session, Course, dan Attendee... tapi nggak ada tombol **Create**, **Delete**, dan **Edit**, karena kita nggak kasi akses create, delete, dan update.

21.2 BIKIN GROUP LEWAT XML

Sekarang kita bikin definisi group melalui XML agar bisa disertakan pada module.

Group yang akan dibuat adalah "Academic / Manager" dan "Academic / User".

Bikin folder `security` di bawah module, bikin file baru namanya `group.xml`, isinya seperti ini.

```
<openerp>
    <data noupdate="0">
        <record id="group_manager" model="res.groups">
            <field name="name">Academic / Manager</field>
        </record>
        <record id="group_user" model="res.groups">
            <field name="name">Academic / User</field>
        </record>
```

```
</data>
</openerp>
Gambar 166 Bikin group via XML
```

Struktur file di folder addons sejauh ini... ada tambahan folder **security** dan file **group.xml**.

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

Tambahin di **__openerp__.py**

```
{
    "name": "Academic Information System Day 4",
    "version": "1.0",
    "depends": [
        "base",
        "account",
        "sale",
    ],
    "author": "akhmad.daniel@gmail.com",
    "category": "Education",
    "website": 'http://www.vitraining.com',
    "description": """
Academic Information System Day 4

""",
    "data": [
        "menu.xml",
        "course.xml",
        "session.xml",
        "attendee.xml",
        "partner.xml",
        "workflow.xml",
        "security/group.xml",
    ],
    "installable": True,
    "auto_install": False,
    "application": True,
}
```

Gambar 167 Panggil **security/group.xml** dari **__openerp__.py**

Restart odoo dan update module.

Hasilnya lihat di **Settings > Users > Groups..** search Adademic.

The screenshot shows the Odoo interface with the 'Settings' tab selected in the top navigation bar. On the left, a sidebar menu is open, showing various sections like 'Dashboard', 'Users', 'Companies', 'General Settings', 'Translations', 'Languages', 'Import / Export', 'Application Terms', and 'Technical'. The 'Groups' section is currently selected and highlighted in blue. The main content area is titled 'Groups' and contains a table with two rows. The first row has a checkbox labeled 'Group Name' and a value 'Academic / Manager'. The second row also has a checkbox labeled 'Group Name' and a value 'Academic / User'. At the top right of the content area, there are several filters: 'Internal Groups', 'Group acad', 'Filters', 'Group By', and 'Favorites'. A caption at the bottom left of the screenshot reads 'Gambar 168 Group list view'.

21.3 MASUKKAN USER KE GROUP

Setelah ini kita udah bisa masukkan User kedalam group Academic / Manager maupun User.

Caranya, masuk ke **Settings > Users > Users**. Pilih user atau create baru.

Klik tab **Access Rights** nya user. Disitu kita bisa tick group di bawah kategori **Other: Academic / Manager** atau **Academic / User**.

Technical Settings

- A warning can be set on a partner (Account)
- Addresses in Sales Orders
- Analytic Accounting for Sales
- Display Incoterms on Sales Order and related invoices
- Manage Pricelist Items
- Manage Product Variants
- Personalize sale order and invoice report
- Pro-forma Invoices
- Sales Pricelists
- Tax display B2C
- A warning can be set on a product or a customer (Sale)
- Analytic Accounting
- Discount on lines
- Manage Multiple Units of Measure
- Manage Product Packaging
- Manage Properties of Product
- Pricelists On Product
- Properties on lines
- Tax display B2B

Extra Rights

- Multi Companies
- Multi Currencies

Other Extra Rights

- Contact Creation
- Public
- Portal

Other

- Academic / Manager
- Session Read
- Academic / User

Gambar 169 Masukkan user ke group Academic/Manager

21.4 IMPORT CSV ACCESS RIGHT

Access right group yang dibuat di module biasanya kita import melalui file CSV. Pada file ini kita definisikan akses read, write, creation dan delete terhadap objec Course, Session dan Attendees untuk group Academic / Manager.

Bikin file baru CSV dengan nama `ir.model.access.csv`
dibawah folder `security`.

Buka di OpenOffice supaya lebih enak editnya... isinya seperti berikut....

The screenshot shows a CSV file titled "attendee_user" with 11 rows of data. The columns are labeled A through H. The data includes XML IDs for models like course_manager, session_manager, attendee_manage, etc., along with their respective group_ids and permission levels (perm_read, perm_write, perm_create, perm_unlink).

A	B	C	D	E	F	G	H
1 id	name	model_id:id	group_id:id	perm_read	perm_write	perm_create	perm_unlink
2 course_manager	course_manager	model_academic_course	group_manager	1	1	1	1
3 session_manager	session_manager	model_academic_session	group_manager	1	1	1	1
4 attendee_manage	attendee_manager	model_academic_attendee	group_manager	1	1	1	1
5 course_user	course_user	model_academic_course	group_user	1	1	1	0
6 session_user	session_user	model_academic_session	group_user	1	1	1	0
7 attendee_user	attendee_user	model_academic_attendee	group_user	1	1	1	0
8 course_all	course_all	model_academic_course		1	0	0	0
9 session_all	session_all	model_academic_session		1	0	0	0
10 attendee_all	attendee_all	model_academic_attendee		1	0	0	0
11							

Gambar 170 File CSV access right

Isi file CSV secara lengkap...

```
"id","name","model_id:id","group_id:id","perm_read","perm_write","perm_create","perm_unlink"
"course_manager","course_manager","model_academic_course","group_manager",1,1,1,1
"session_manager","session_manager","model_academic_session","group_manager",1,1,1,1
"attendee_manage","attendee_manager","model_academic_attendee","group_manager",1,1,1,1
"course_user","course_user","model_academic_course","group_user",1,1,1,0
"session_user","session_user","model_academic_session","group_user",1,1,1,0
"attendee_user","attendee_user","model_academic_attendee","group_user",1,1,1,0
"course_all","course_all","model_academic_course",,1,0,0,0
"session_all","session_all","model_academic_session",,1,0,0,0
"attendee_all","attendee_all","model_academic_attendee",,1,0,0,0
```

File CSV ini terdiri dari beberapa kolom sesuai ketentuan odoo yaitu:

Kolom **id**, adalah nama reference XML ID

Kolom **name**, adalah nama XML record

Kolom **model_id:id**, nama model yang mau dikasi hak aksesnya kepada suatu group dengan format

model_<module>_<classname>.

Kolom **group_id:id**, nama group XML ID yang udah diinsert melalui **group.xml** yaitu group yang diberi hak akses terhadap module. Kalau dikosongin berarti berlaku untuk semua group yang ada

Kolom `perm_read`, akses read boleh (1) atau nggak (0)

Kolom `perm_write`, akses update boleh (1) atau nggak (0)

Kolom `perm_create`, akses bikin record baru boleh (1) atau nggak (0)

Kolom `perm_unlink`, akses delete record boleh (1) atau nggak (0)

Lanjut, tambahi di `__openerp__.py`

```
{  
    "name": "Academic Information System Day 4",  
    "version": "1.0",  
    "depends": [  
        "base",  
        "account",  
        "sale",  
    ],  
    "author": "akhmad.daniel@gmail.com",  
    "category": "Education",  
    'website': 'http://www.vitraining.com',  
    "description": """\nAcademic Information System Day 4  
""",  
    "data": [  
        "menu.xml",  
        "course.xml",  
        "session.xml",  
        "attendee.xml",  
        "partner.xml",  
        "workflow.xml",  
        "security/group.xml",  
        "security/ir.model.access.csv",  
    ],  
    "installable": True,  
    "auto_install": False,  
    "application": True,  
}
```

Gambar 171 Panggil `security/ir.model.access.csv` dari `__openerp__.py`

Sejauh ini struktur folder addons kita adalah...

```
academic  
|-- __init__.py  
|-- __openerp__.py  
|-- attendee.py  
|-- attendee.xml  
|-- course.py  
|-- course.xml
```

```

|-- menu.xml
|-- partner.py
|-- partner.xml
|-- security
|   |-- group.xml
|   `-- ir.model.access.csv
|-- session.py
|-- session.xml
`-- workflow.xml

```

Restart odoo dan update module.

Hasilnya lihat di **Settings > Users > Groups.. search Adademic**. Klik Group Manager, klik tab **Access Rights**. Isinya harus sesuai dengan yang didefinisikan pada file CSV.

The screenshot shows the Odoo Groups interface. At the top, there are buttons for 'Edit' and 'Create', and a 'Action' dropdown. On the right, it says '3 / 37'. The main area displays a table titled 'Access Rights' for the 'Academic / Manager' group. The table has columns for 'Object', 'Read Access', 'Write Access', 'Create Access', 'Delete Access', and 'Name'. Three objects are listed: 'academic.course', 'academic.session', and 'academic.attendee'. For each object, all access rights (Read, Write, Create, Delete) are checked. The 'Name' column lists 'course_manager', 'session_manager', and 'attendee_manager' respectively. There are also tabs for 'Users', 'Inherited', 'Menus', 'Views', 'Rules', and 'Notes'.

Object	Read Access	Write Access	Create Access	Delete Access	Name
academic.course	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	course_manager
academic.session	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	session_manager
academic.attendee	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	attendee_manager

Gambar 172 Access right group Manager

Klik group User, klik tab Access Right. Isinya harus sesuai dengan yang didefinisikan pada file CSV.

Object	Read Access	Write Access	Create Access	Delete Access	Name
academic.course	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	course_user
academic.session	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	session_user
academic.attendee	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	attendee_user

Gambar 173 Access right group User

21.5 RECORD RULES

Pada awalnya diatas, semua Group Manager bisa edit dan delete Course. Kita mau hanya yang responsible aja yang bisa delete dan update. Untuk itu perlu dibuat Record Rules.

Record rule dipakai untuk memberi access rights hanya kepada beberapa records suatu model oleh suatu group. Misalnya data Customer, semua group Sales boleh akses read, create, write, dan delete. Tapi kemudian kita batasi hanya Sales User yang berada pada Area yang sama dengan Customer yang boleh read, write, dan delete. Sales User pada Area yang berbeda nggak bisa lihat Customer itu.

Record Rule merupakan record dari model `ir.rule`, dan dikaitkan ke model, beberapa groups (many2many field), permissions yang terkait, dan suatu kriteria domain yang menentukan records yang mana aja access rights diperbolehkan untuk group tersebut.

Sekarang kita tambahin Record Rule untuk model Course pada group `OpenAcademy / Manager`, yang membatasi akses “write” and “unlink” / delete hanya kepada siapa user yang

responsible atas Course tersebut. Jika Course belum ada responsible-nya, semua user pada group itu boleh mengupdate dan delete.

Buka lagi file `security/groups.xml`, tambahi rule record seperti ini...

```
<odoo>
<data noupdate="0">
    <record id="group_manager" model="res.groups">
        <field name="name">Academic / Manager</field>
    </record>
    <record id="group_user" model="res.groups">
        <field name="name">Academic / User</field>
    </record>

    <record id="only_responsible_can_modify" model="ir.rule">
        <field name="name">Cuman penanggung jawab yang
            bisa edit Course</field>
        <field name="model_id" ref="model_academic_course"/>
        <field name="groups" eval="[(4, ref('group_manager'))]"/>
        <field name="perm_read" eval="0"/>
        <field name="perm_write" eval="1"/>
        <field name="perm_create" eval="0"/>
        <field name="perm_unlink" eval="1"/>
        <field name="domain_force">['|', ('responsible_id','=',False),
            ('responsible_id','=',user.id)]</field>
    </record>
</data>
</odoo>
```

Gambar 174 Bikin Record Rule dari XML

Disini kita tambahi record untuk object `ir.rule`. Field record ini adalah:

Field `name` untuk label nama record rule.

Field `model_id` adalah reference ke nama model dalam format `model_<module>_class`, contohnya `model_academic_course`.

Field `groups` adalah relasi many2many ke groups, dimana penginputannya harus menggunakan atribut eval yang me-refer ke id group manager yang ditulis pada file XML `group.xml` sebelumnya, seperti ini :

```
[ (4, ref('academic.group_manager')) ]
```

Field `perm_read`, `perm_write`, `perm_create`, dan `perm_unlink` menentukan apakah boleh (1) atau tidak (0) untuk read, write (update), create, dan unlink (delete).

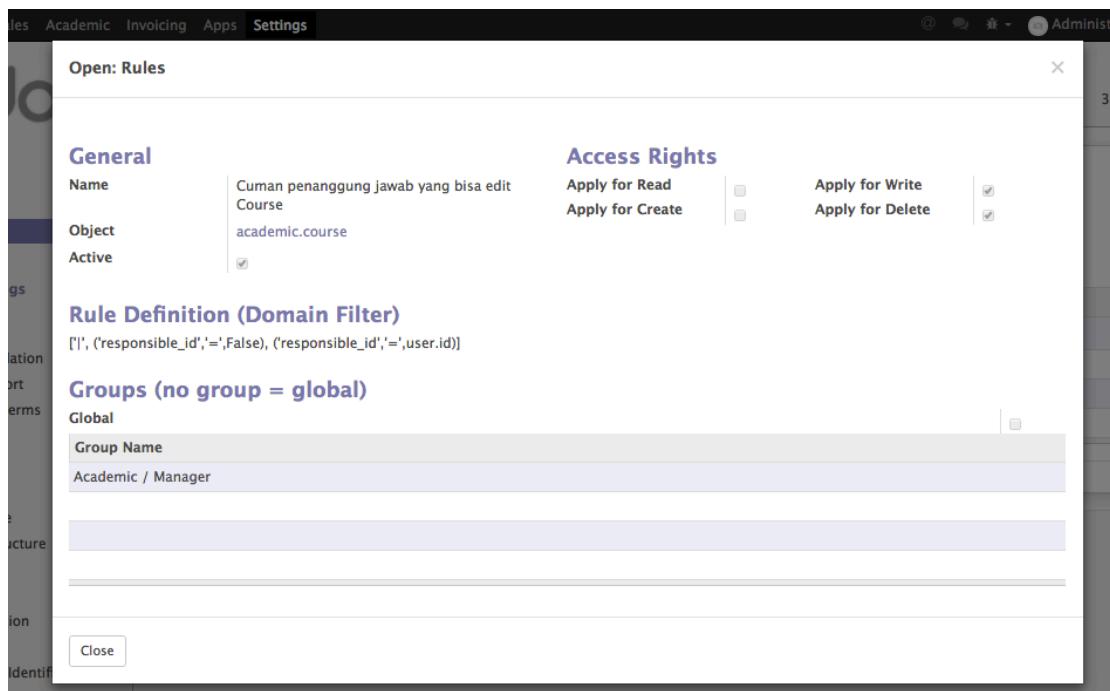
Field `domain_force` isinya filter domain yang membatasi record mana yang boleh diakses hanya oleh group yang terdaftar pada field `groups`. Jika domain ini nggak terpenuhi maka record ini nggak bisa diakses.

Restart odoo dan update module, hasilnya...

The screenshot shows the Odoo Groups interface. At the top, it says "Groups / Academic / Manager". Below that are buttons for "Edit" (highlighted in blue), "Create", "Action", and "3 / 37". The main area has tabs for "Application", "Portal", "Users", "Inherited", "Menus", "Views", "Access Rights", "Rules" (which is selected and highlighted in blue), and "Notes". Under the "Rules" tab, there is a table with one row. The table has columns for "Name", "Object", and "Global". The row contains the text "Cuman penanggung jawab yang bisa edit Course" in the "Name" column, "academic.course" in the "Object" column, and an unchecked checkbox in the "Global" column.

Gambar 175 Ada record rules di group Academic/Manager

Klik detail Record rule, muncul Access Rights, Groups, dan Domain Filter sesuai file XML.



Gambar 176 Isi Record rules

Sekarang, coba kita login dengan user yang bukan responsible dari suatu Course, lalu coba update dan delete...

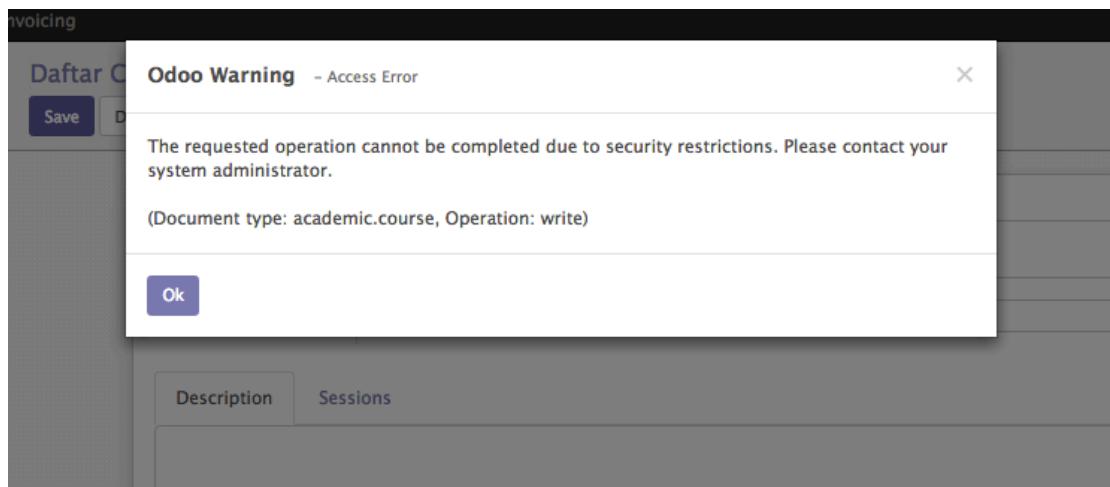
Misalkan kondisinya sekarang kayak gini..

Daftar Course			
	Name	Description	Responsible
<input type="checkbox"/>	PHP		Agus
<input type="checkbox"/>	Java		Agus
<input type="checkbox"/>	Odoo		Badu
<input type="checkbox"/>	Dot Net		Badu

Gambar 177 Responsible user masing-masing Course

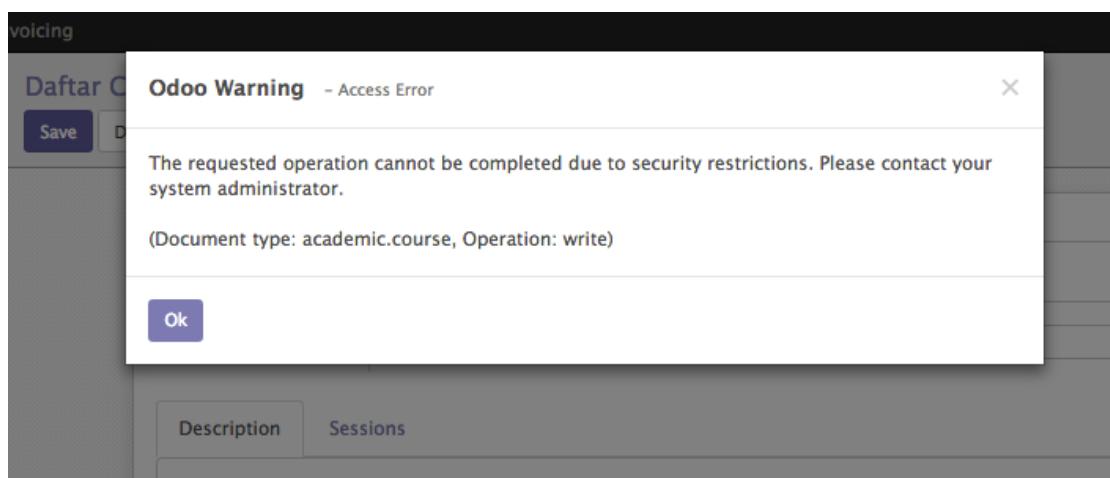
User agus dan badu udah diset sebagai group Academic / Manager.

Apakah agus bisa edit / delete Course-nya badu, Odoo dan Dotnet?



Gambar 178 Agus nggak bisa edit course-nya Badu

Apakah **badu** bisa edit / delete Course-nya **agus**, PHP dan Java?



Gambar 179 Badu nggak bisa delete Course-nya Agus

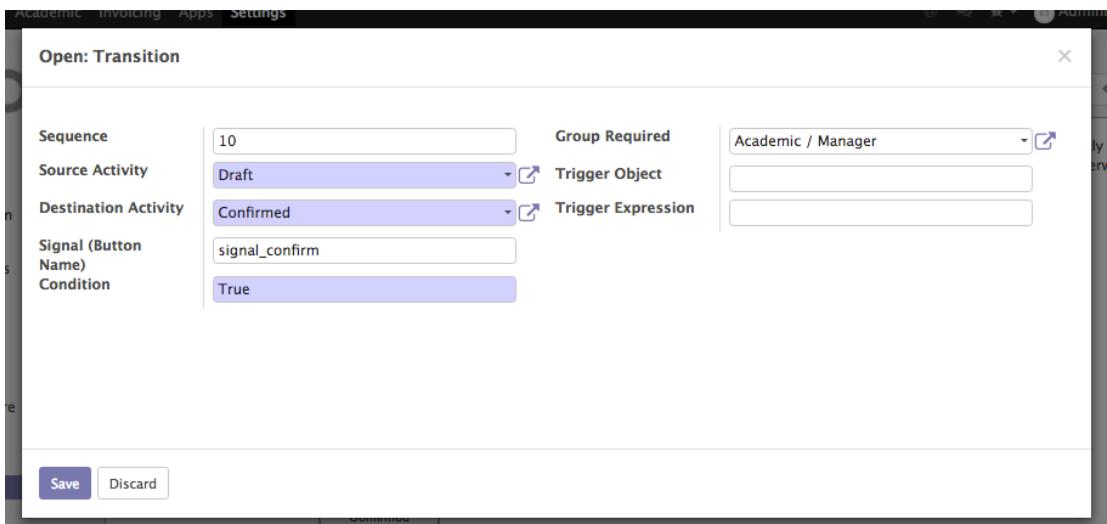
21.6 HUBUNGAN GROUP WORKFLOW

Hubungan workflow dengan object adalah meng-update field **state**, jadi untuk bisa menjalankan workflow, user harus punya akses write terhadap object tersebut.

Coba kurangi access Write terhadap group User. Otomatis group User tidak bisa menjalankan workflow. Kalau dia klik tombol workflow akan terjadi error Access Denied.

Pada workflow action (panah) boleh dipilih nama group yang bisa men-trigger signal ini misalnya group Manager. Sehingga

kalau tombol workflow diklik oleh group User, tidak terjadi error karena signal tidak ter trigger.



Gambar 180 Set Group yang bisa trigger signal workflow

Lalu, bagaimana meng-hide tombol sesuai group yang login?

Untuk ini kita perlu tambahi atribut `groups` pada tombol supaya hanya muncul kalau user yang login tergabung pada group yang ditentukan disitu.

Edit file `session.xml`.

Tambahkan atribut `groups` pada button dengan nama-nama group yang boleh melihat :

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

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

Gambar 181 Modif tombol supaya muncul sesuai group

Akibatnya, hanya user dalam group Academic/ Manager yang bisa melihat tombol-tombol workflow.

22 WIZARD

Di odoe ada yang namanya wizard, gunanya untuk memudahkan kita dalam mengisi data atau memilih parameter suatu proses berikutnya.

Misalnya untuk report kita bisa pilih terlebih dahulu parameter report misalnya periode, account, tahun, dan lain-lain. Ketika Submit maka akan diproses report sesuai dengan parameter yang dipilih pada wizard.

Wizard bisa juga untuk memudahkan kita dalam men-entry data. Misalnya disini kita buat wizard untuk menambahkan Attendee ke suatu Session tapi nggak satu-per-satu. Jadi kita bisa input banyak Attendee sekaligus kedalam suatu Session.

22.1 DEFINISIKAN CLASS WIZARD

Bikin directory baru namanya **wizard** dibawah folder addons
academic.

Bikin file baru `create_attendee.py` dibawah folder `wizard`.

Isinya ada 2 class, yaitu `CreateAttendeeWizard` dan `AttendeeWizard`.

Gambar 182 Bikin class wizard

Class `CreateAttendeeWizard` adalah class wizardnya itu sendiri, yang gunanya untuk membentuk window pop up wizard, dan terdiri dari kolom-kolom sesuai definisi pada class itu.

Class ini mirip banget seperti class object odoo lainnya, tapi disini kita pakai `models.TransientModel` sebagai parent class nya. Class `models.TransientModel` mirip seperti `models.Model` hanya saja datanya nggak tersimpan ke table database, tapi di memory.

Class ini punya 2 kolom yaitu `session_id` (pilihan Session yang Attendee nya mau di input), dan `attendee_ids` yaitu data calon Partner yang akan mengikuti Session yang dipilih.

Class berikutnya, `AttendeeWizard` adalah untuk menampung data Partner sementara yang dipilih pada wizard yang nantinya akan dimasukkan sebagai attendee session.

Ini diperlukan karena Partner yang dipilih disini belum masuk ke class Attendee yang di database, tapi masih ditampung dulu sementara di memory. Relasinya ke wizard class adalah many2one. Artinya satu wizard bisa punya banyak `AttendeeWizard`. Class ini juga punya relasi many2one ke Partner.

Lanjut, bikin file baru namanya `__init__.py` dibawah directory `wizard`:

```
import create_attendee
```

Gambar 183 Import class wizard

Struktur directory addons kita sejauh ini adalah...

```
academic
|-- __init__.py
|-- __openerp__.py
|-- attendee.py
```

```
|-- attendee.xml
|-- course.py
|-- course.xml
|-- menu.xml
|-- partner.py
|-- partner.xml
|-- security
|   |-- group.xml
|   `-- ir.model.access.csv
|-- session.py
|-- session.xml
|-- wizard
|   |-- __init__.py
|   `-- create_attendee.py
`-- workflow.xml
```

Lanjut, modif file `__init__.py` yang ada di bawah folder `academic`, tambahi `import wizard` artinya kita mengimport satu folder `wizard` beserta isinya sekaligus.

```
import wizard
```

Gambar 184 Import folder wizard

22.2 BIKIN MENU UNTUK WIZARD

Pertama-tama kita bikin menu item dan action yang diperlukan untuk membuka wizard. Buat file baru di dalam directory `wizard`, file XML dengan nama `create_attendee_view.xml` yang berisi menu item dan action window `create_attendee_wizard_action`:

```
<openerp>
    <data>
        <record model="ir.actions.act_window"
            id="create_attendee_wizard_action">
            <field name="name">Add attendee</field>
            <field name="res_model">academic.create.attendee.wizard</field>
            <field name="view_type">form</field>
            <field name="view_mode">form</field>
            <field name="target">new</field>
        </record>

        <menuitem name="Add Attendee" parent="academic_1"
            id="create_attendee_wizard_menu"
            sequence="40"
            action="create_attendee_wizard_action"/>
    </data>
</openerp>
```

Gambar 185 Menu dan action window wizard

Wizard dijalankan melalui record `ir.actions.act_window` alias action window, tapi ditambahi field `target` benilai new. Field ini gunanya untuk membuka wizard view pada popup window. Field `res_model` isinya nama model wizard yang akan dibuka.

Action window diatas di-trigger lewat menu item yang parent nya adalah main menu `academic_1`.

Struktur file addons kita harus jadi seperti ini

```
academic/
|-- __init__.py
|-- __openerp__.py
|-- attendee.py
|-- attendee.xml
|-- course.py
|-- course.xml
|-- menu.xml
|-- partner.py
|-- partner.xml
|-- security
|   |-- group.xml
|   `-- ir.model.access.csv
|-- session.py
|-- session.xml
|-- wizard
|   |-- __init__.py
|   |-- create_attendee.py
|   `-- create_attendee_view.xml
`-- workflow.xml
```

Lanjut, update `__openerp__.py` yang berada di bawah folder `academic`:

```
{
    "name": "Academic Information System Day 4",
    "version": "1.0",
    "depends": [
        "base",
        "account",
        "sale",
    ],
    "author": "akhmad.daniel@gmail.com",
    "category": "Education",
    "website": 'http://www.vitraining.com',
    "description": """
Academic Information System Day 4
* Add security groups
* Add security ir.model.access.csv
* Add wizard
"""
,
    "data": [
```

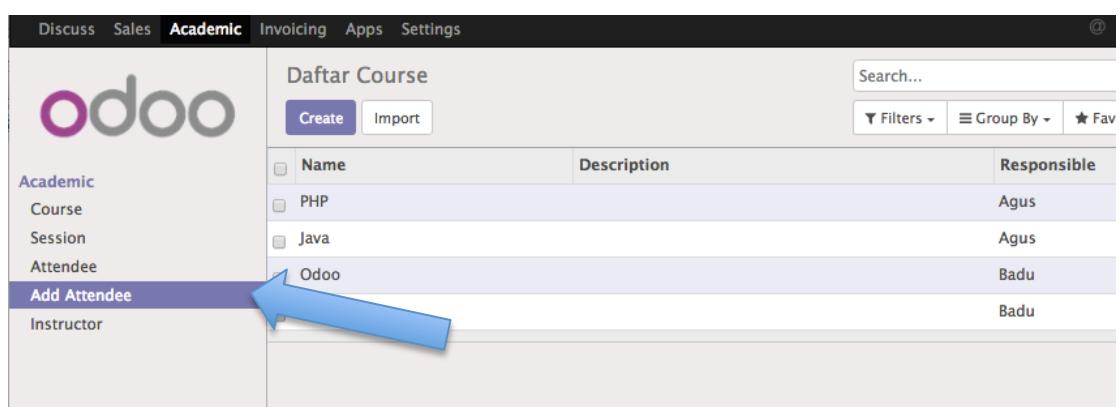
```

    "menu.xml",
    "course.xml",
    "session.xml",
    "attendee.xml",
    "partner.xml",
    "workflow.xml",
    "security/group.xml",
    "security/ir.model.access.csv",
    "wizard/create_attendee.xml",
],
"installable": True,
"auto_install": False,
"application": True,
}

```

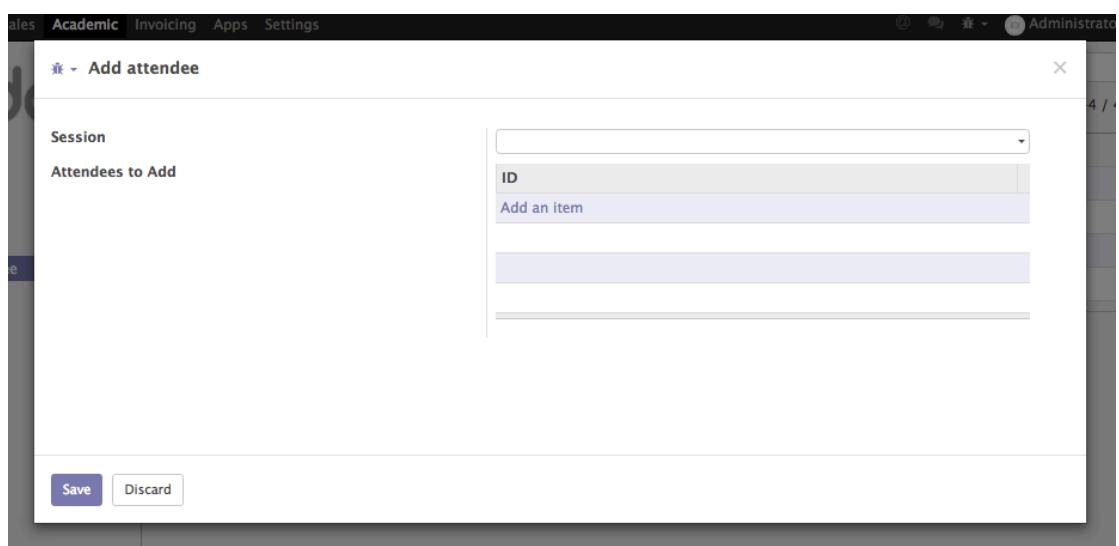
Gambar 186 Panggil wizard/create_attendee_view.xml dari __openerp__.py

Restart odoo dan update module, hasilnya...



Gambar 187 Menu Add attendee muncul

Ada menu baru, **Add Attendee**, waktu di-click menu tersebut akan muncul pop up box seperti ini...



Gambar 188 Wizard pop up muncul

Okee, menu dan wizard udah muncul waktu menu di-klik.

Tapi tampilannya masih belum betul.. kita modif abis ini...

22.3 MODIF FORM VIEW WIZARD

Edit file `create_attendee_view.xml`.

Tambahi record form view supaya nampilin semua fields class wizard. Lalu tambahi dua button pada form view yaitu untuk submit dan cancel wizard.

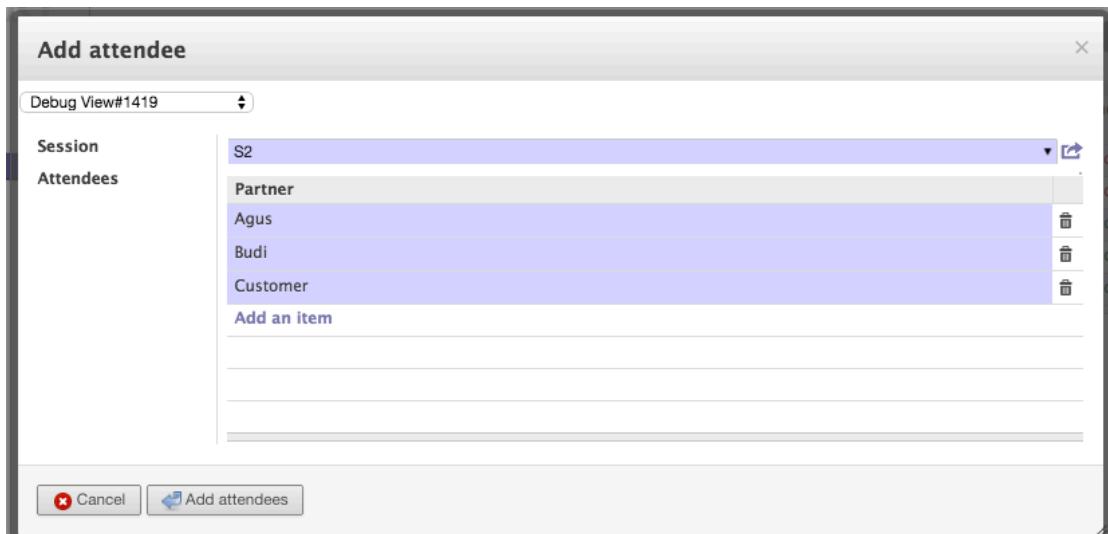
```
<record model="ir.ui.view" id="create_attendee_form_view">
    <field name="name">academic.create.attendee.wizard.form</field>
    <field name="model">academic.create.attendee.wizard</field>
    <field name="type">form</field>
    <field name="arch" type="xml">
        <form string="Add attendee" version="7.0">
            <group>
                <field name="session_id"/>
                <field name="attendee_ids" mode="tree">
                    <tree string="Attendees" editable="bottom">
                        <field name="partner_id"/>
                    </tree>
                </field>
            </group>
            <footer>
                <button type="special"
                    special="cancel"
                    string="Cancel"
                    icon="fa-times"/>

                <button type="object"
                    name="action_add_attendee"
                    string="Add attendees"
                    icon="fa-check-circle"
                    confirm="Are you sure you want to add those attendees?"
                    />
            </footer>
        </form>
    </field>
</record>
```

Gambar 189 Modif form view wizard dan tambah button

Letakkan dibawah definisi menuitem yang udah dibuat sebelumnya...

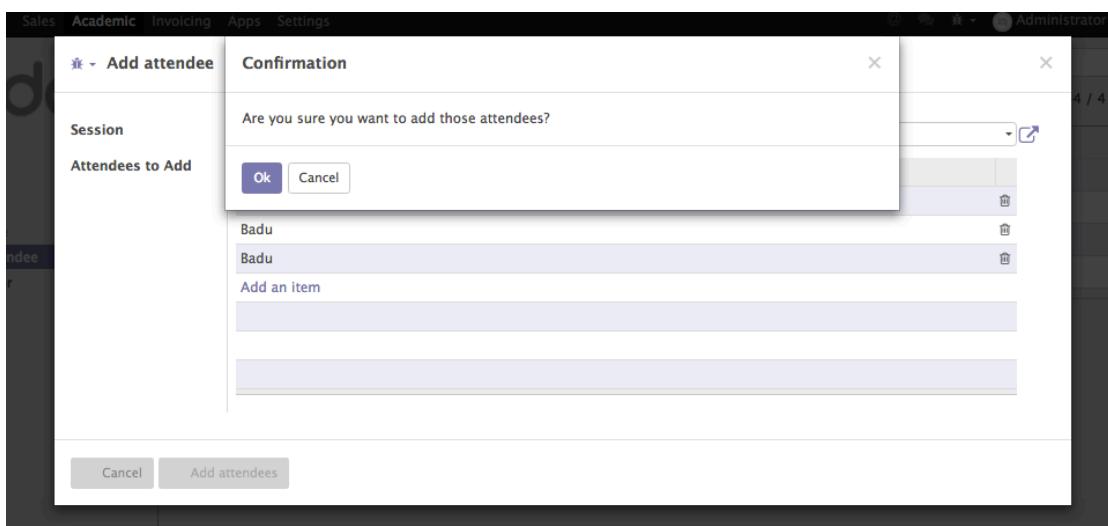
Restart odoo dan update module, hasilnya...



Gambar 190 Form view wizard udah lengkap

Semua field yang kita butuhkan udah muncul, termasuk tombol submit dan cancel.

Waktu click tombol submit Add Attendees...



Gambar 191 Konfirmasi submit wizard

Udah ada respon, tapi belum ada efek apa-apa karena kita belum bikin fungsi untuk nangkep action dari wizard... yaitu `action_add_attendee` sesuai dengan attribute `name` button yang ber-type object.

22.4 BIKIN METHOD UNTUK MEMPROSES DATA WIZARD

Edit file `create_attendee.py`.

Buat method baru, namanya `action_add_attendee` di dalam class, seperti ini.

```
class CreateAttendeeWizard(models.TransientModel):
    _name = 'academic.create.attendee.wizard'

    session_id = fields.Many2one(comodel_name="academic.session",
                                 string="Session", required=False, )
    attendee_ids = fields.One2many(comodel_name="academic.attendee.wizard",
                                   inverse_name="wizard_id",
                                   string="Attendees to Add",
                                   required=False, )

    # create method
@api.multi
def action_add_attendee(self):
    self.ensure_one()
    session = self.session_id
    att_data = [{'partner_id': att.partner_id.id}
                for att in self.attendee_ids]
    session.attendee_ids = [(0, 0, data) for data in att_data]

    return {'type': 'ir.actions.act_window_close'}
```

Gambar 192 Method proses add attendee

Pada method ini ada parameter bawaan odoo yaitu `self` yang berisi object record wizard. Pertama kita pastikan bahwa `self` hanya berisi 1 record dengan:

```
self.ensure_one()
```

Dari variable record wizard `self` tersebut, kita bisa ambil data `session_id` dan `attendee_ids` yang dipilih pada wizard, yaitu `self.session_id` dan `self.attendee_ids`.

Attribut `self.attendee_ids` berisi array list `partner_id`, dan inilah yang perlu kita masukkan ke database melalui object `academic.attendee`.

Lanjut, bentuk array list `att_data` melalui list comprehension

```
att_data = [{'partner_id': att.partner_id.id}
            for att in self.attendee_ids]
```

artinya, pada setiap elemen yang ada pada list `self.attendee_ids` simpan ke local variable `att`, lalu bentuk menjadi dictionary `{'partner_id': att.partner_id.id}`.

Hasil list comprehension diatas, att akan menajadi array list of dictionary, misalnya:

```
[ { 'partner_id' : 3 } , {'partner_id':2} , {'partner_id':20} ]
```

udah itu, kita jalankan penulisan field `attendee_ids` pada pada object session untuk mengupdate field `attendee_ids` nya.

```
session.attendee_ids = [(0, 0, data) for data in att_data]
```

Untuk mengupdate field one2many membutuhkan format data khusus yang disebut one2many commands. Dalam hal ini kita mengupdate field `attendee_ids` dalam rangka untuk nambahin record, jadi perlu one2many command insert.

Disini lagi-lagi kita gunakan list comprehension untuk membentuk array list dengan format data one2many commands:

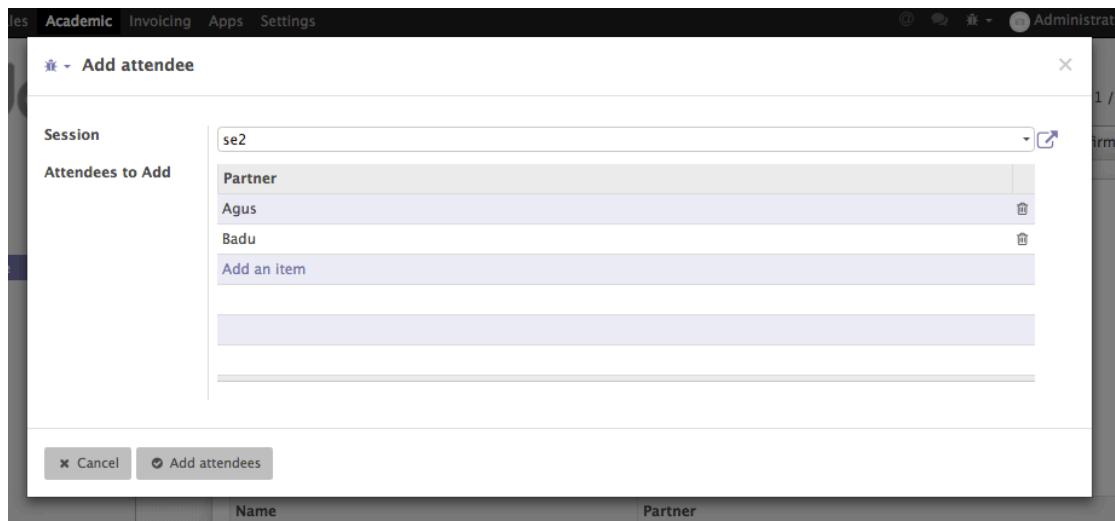
```
[  
    (0,0, {'partner_id': 3} ) ,  
    (0,0, {'partner_id': 2} ) ,  
    (0,0, {'partner_id': 20} )  
]
```

Sumber data yang kita looping pada list comprehension adalah `att_data` yang udah disiapkan di atas sebelumnya.

Setiap element `att_data` kita simpan kedalam variable local `data`, yang kemudian dibentuk menjadi tuple `(0,0,data)` sehingga terbentuklah array list seperti yang diinginkan di atas.

Setelah terbentuk dengan rapih maka boleh dijadikan parameter untuk nilai field one2many `attendee_ids` pada session object.

Update module dan jalankan wizard.



Gambar 193 Add attendees dari wizard

Setelah klik tombol Add attendees, dan konfirmasi maka attendees akan masuk ke session yang dipilih.

The screenshot shows the 'Daftar Session / se2' page. At the top, there are buttons for 'Edit' and 'Create', and a 'Draft' status indicator. The main area displays session details: 'se2', 'Course' (empty), 'Instructor' (empty), 'Start Date' (02/21/2017), 'Image Small' (camera icon), 'Duration' (0), 'Seats' (0), 'Active' (checked), 'Taken Seat' (0%), and '0%' completion. Below this is a table titled 'Attendees' with columns 'Name' and 'Partner'. It lists 'Agus' and 'Badu' under the 'Partner' column.

Gambar 194 Hasil add attendee melalui wizard

22.5 BIKIN MENU CONTEXT

Ada cara lain untuk membuka wizard yaitu melalui record `ir.actions.act_window` seperti diatas, tapi ditambahi field `src_model`.

Field ini menentukan pada konteks object manakah action ini tersedia. Menu Wizard akan muncul pada contextual actions model yaitu di bawah tombol More pada form view. Karena ada hal internal yang perlu dilakukan khusus untuk ini, action context kita diklarasikan melalui XML tag `act_window`.

Disini kita akan mengaitkan wizard dengan menu context session model. Kita akan menggunakan argumen “context” untuk mendefinisikan session yang sedang dibuka sebagai nilai default untuk field `session_id` di wizard.

22.5.1 EDIT FILE WIZARD/CREATE_ATTENDEE_VIEW.XML.

Tambahi XML element yang dibutuhkan untuk membuat action context.

```
<act_window id="session_create_attendee_wizard"
            name="Add Attendees"
            src_model="academic.session"
            res_model="academic.create.attendee.wizard"
            view_mode="form"
            target="new"
            key2="client_action_multi"/>

</data>
</openerp>
```

Gambar 195 Action window untuk context menu

Field `src_model` menentukan pada object manakan action context ini akan muncul di form view nya.

Field `res_model` menentukan nama lengkap dari object tersebut.

Field `view_mode` menentukan mode view, yaitu form view.

Field `target` isinya new agar muncul pop up window.

Field `key2` menentukan bagaimana data context akan berisi apakah multi id atau single id.

22.5.2 EDIT FILE WIZARD/CREATE_ATTENDEE.PY

Pada class `CreateAttendeeWizard`, definisikan default value untuk field `session_id` yang nilainya diambil dari form view session yang sedang dibuka melalui variable context.

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

class CreateAttendeeWizard(models.TransientModel):
    _name = 'academic.create.attendee.wizard'

    def _get_active_session(self):
        context = self.env.context
        if context.get('active_model') == 'academic.session':
            return context.get('active_id', False)
        return False

    session_id = fields.Many2one(comodel_name="academic.session",
                                 string="Session", required=False,
                                 default=_get_active_session)
    attendee_ids = fields.One2many(comodel_name="academic.attendee.wizard",
                                   inverse_name="wizard_id",
                                   string="Attendees to Add",
                                   required=False, )
```

Gambar 196 Cara ambil active session

Atribut `default`s untuk field `session_id` kita definisikan melalui method `_get_active_session()`.

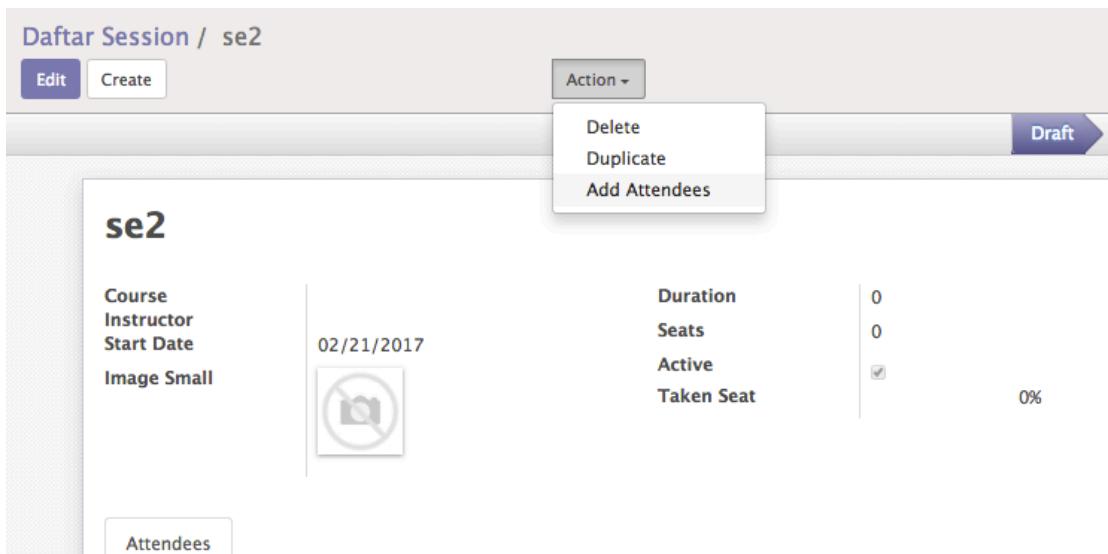
Pada method ini kita membaca variable context yang datang dari action menu. Di dalam context itu tersedia key `active_model` dan `active_id` disamping key lainnya yang belum kita perlukan sekarang.

Disini kita cuma cek apakah key `active_model` adalah betul `academic.session`. jika ya, maka ambil key `active_id` dan jadikan

return value method ini sehingga masuk nantinya sebagai default value untuk field **session_id**.

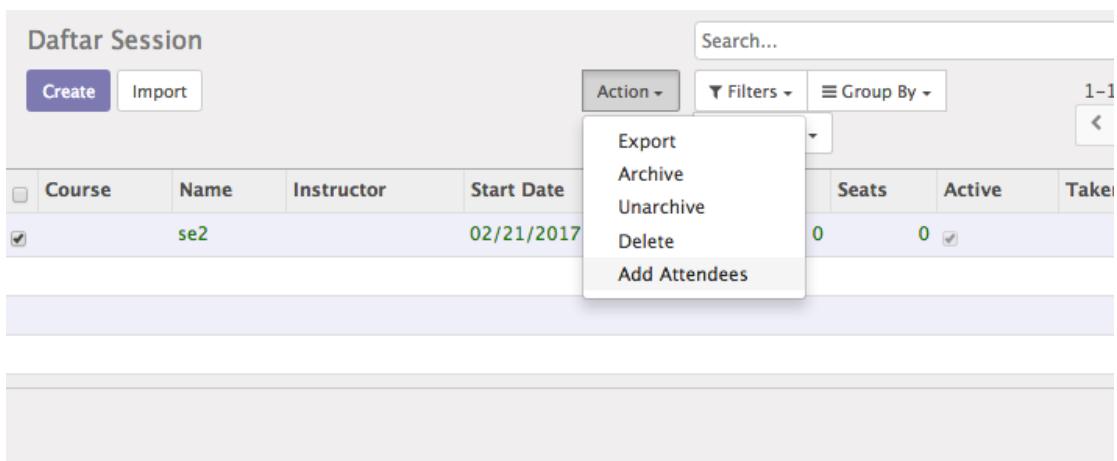
Restart odoo dan update module.

Hasilnya muncul menu Add Attendees dibawah context menu More ... di halaman form Session.



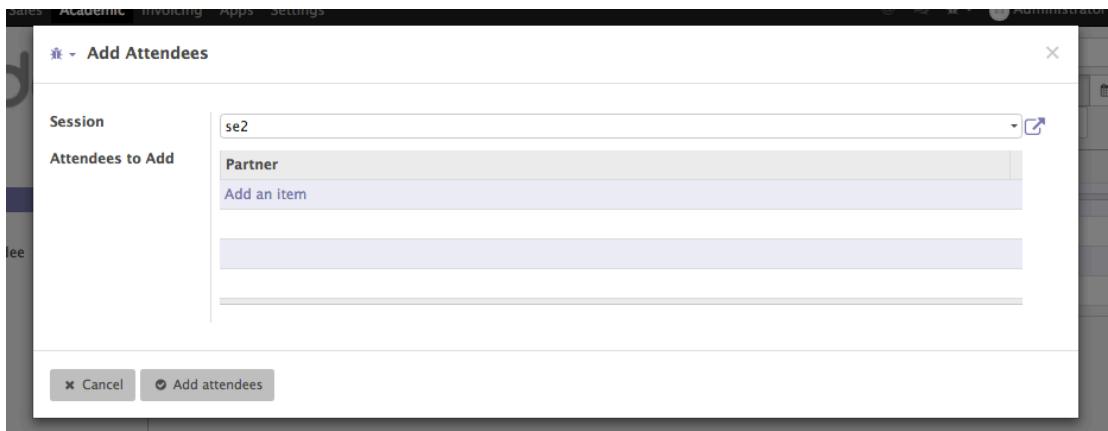
Gambar 197 Muncul context menu di form view Session

dan juga di list **Session** waktu dipilih salah satunya...



Gambar 198 Muncul context menu di list view ketika dipilih salah satu session

Waktu menu **Add Attendee** diclick, wizard otomatis muncul dan session udah terisi sesuai yang dipilih sebelumnya.



Gambar 199 Default value Session di wizard

22.6 ADD AN ONCHANGE METHOD

Quiz: tambahin onchange method untuk menampilkan attendance yang sudah ada pada session yang dipilih pada wizard.

22.7 WIZARD UNTUK BANYAK SESSION SEKALIGUS

Gimana kalau wizardnya harus bisa nambahin banyak Attendee ke banyak Session sekaligus ?

22.7.1 EDIT FILE WIZARD/CREATE_ATTENDEE.PY

Modif model wizard, ganti field "session_id" yang tadinya many2one jadi field many2many "session_ids".

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

class CreateAttendeeWizard(models.TransientModel):
    _name = 'academic.create.attendee.wizard'

    def _get_active_session(self):
        context = self.env.context
        if context.get('active_model') == 'academic.session':
            return context.get('active_ids', False)
        return False

    session_ids = fields.Many2many(comodel_name="academic.session",
        string="Sessions", )

    attendee_ids = fields.One2many(comodel_name="academic.attendee.wizard",
        inverse_name="wizard_id", string="Attendees to Add",
        required=False, )
```

Gambar 200 Field session_ids many2many

Modif juga method default valuenya supaya menyesuaikan dengan field `session_ids`...

Disini method `_get_active_sessions()` mengambil key `active_ids` dari `context` variable dan bukan `active_id` seperti sebelumnya. Hasilnya berupa list of ids dari session yang dipilih di wizard.

Nilai default `session_ids` diisikan dengan return value dari function tersebut.

Apalagi method `action_add_attendee` harus juga dimodif karena data `session_ids` yang datang dari wizard bukan berupa integer lagi, tapi udah array list dari banyak session..

```
@api.multi
def action_add_attendee(self):
    self.ensure_one()
    sessions = self.session_ids
    att_data = [{'partner_id': att.partner_id.id}
                for att in self.attendee_ids]

    for session in sessions:
        session.attendee_ids = [(0, 0, data) for data in att_data]

    return {'type': 'ir.actions.act_window_close'}
```

Gambar 201 Cara save ke session attendee_ids

Variable `sessions` akan berisi list dari id yang dipilih pada wizard.

List ini kita looping satu per satu, lalu pada setiap element list yanb berupa object sesssion, dilakukan update terhadap field `attendee_ids` seperti cara sebelumnya.

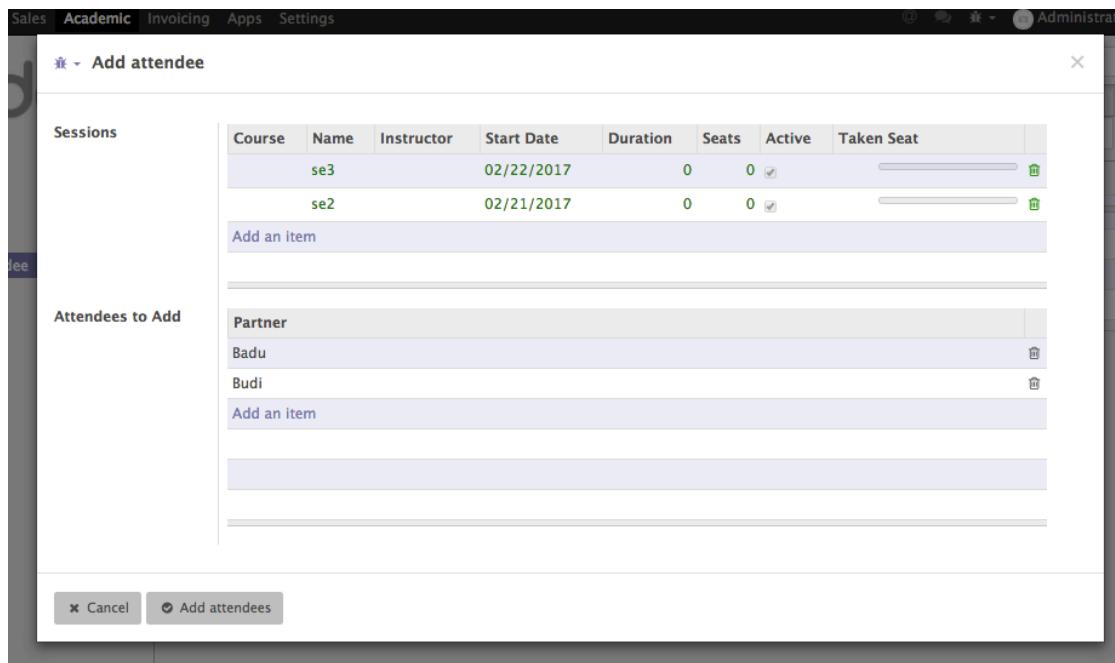
22.7.2 EDIT FILE XML WIZARD/CREATE_ATTENDEE.XML

Ganti pemanggilan field `session_id` menjadi `session_ids`.

```
<field name="arch" type="xml">
<form string="Add attendee" version="7.0">
    <group>
        <field name="session_ids"/>
        <field name="attendee_ids" mode="tree">
            <tree string="Attendees"
                  editable="bottom">
                <field name="partner_id"/>
```

```
</tree>
</field>
</group>
```

Restart odoo dan update module, hasilnya



Gambar 202 Bisa insert banyak attendee sekaligus ke banyak session

23 REKAP HARI 4

Wuiiihhh makin seru nih gan... udah hari ke-4. Mudah-mudahan bukan makin pusing 😊

Berikut rekap apa yang udah kita pelajari di hari ke 4 ini.

Security

Bikin group, access right group, masukkan user ke group, import CSV untuk access right, access menu, record rules, dan security workflow.

Wizard

Membuat wizard class, modifikasi form wizard, method untuk memproses data wizard, context menu, on change di wizard, many2many relation.

24 HARI 5: INTERNATIONALIZATION

Setiap module bisa punya translasi bahasa yang berada di direktori `i18n`, supaya kalau odoo digunakan oleh user menggunakan bahasa tertentu, maka bahasa modul kita juga ikut berubah sesuai bahasa yang dipilih.

Direktori ini berisi file kamus bahasa bernama `LANG.po` dimana `LANG` adalah locale code bahasa yang mau disediakan. Bisa juga berupa kombinasi bahasa dan negara kalo berbeda (misalnya `pt.po` atau `pt_BR.po` untuk bahasa Portugis dan Protugis di Brazil).

Translasi akan di-load otomatis oleh odoo untuk semua bahasa yang disediakan oleh module.

Developer harus pake bahasa English waktu membuat modul, lalu export semua istilah (label, nama field, dll) yang ada pada module pake fitur export gettext POT untuk membentuk file template POT, lalu meng-copy PO files yang udah diexport ke dalam folder `i18n` module.

24.1 BIKIN DIREKTORI `i18n`

Bikin dulu direktori `i18n` dibawah direktori addons academic.

24.2 BAHASA UDAH ADA DI ODOO

24.2.1 INSTALL BAHASA TARGET

Kalo bahasa yang mau kita translate udah ada di odoo, maka instal dulu bahasa itu melalui menu odoo:

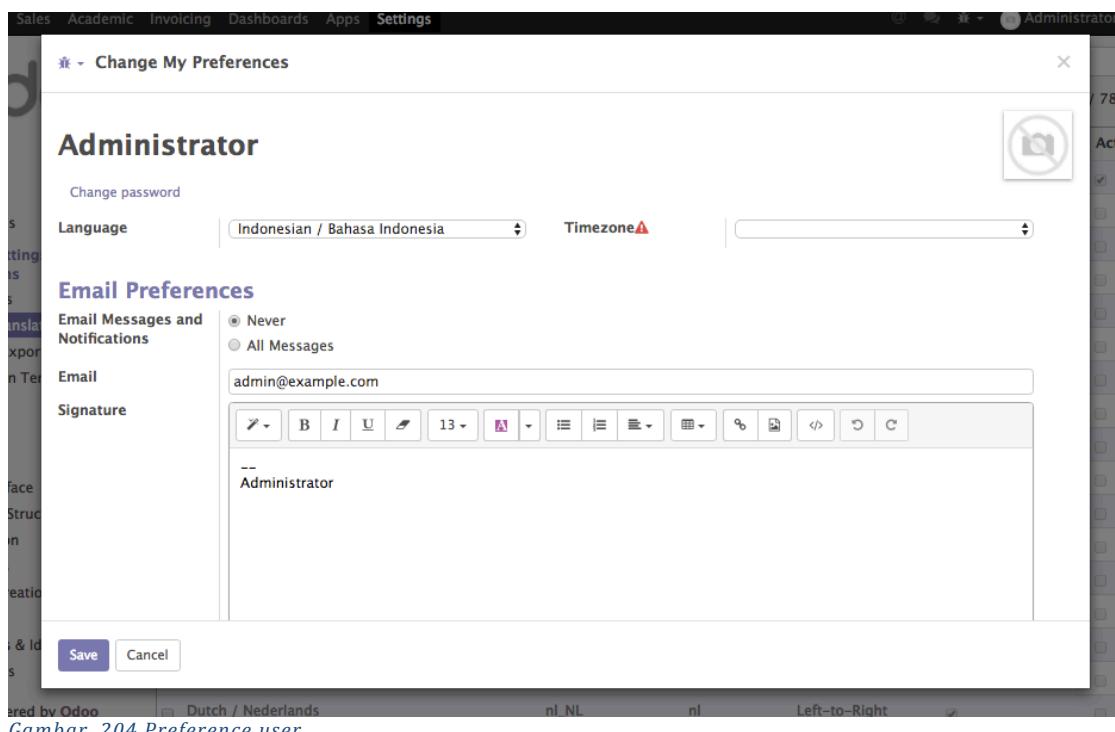
Klik menu `Settings > Translations > Load a Translation`

Pilih Bahasa Indonesia.



Gambar 203 Load Transalction Bahasa Indonesia

Bahasa Indonesia siap digunakan oleh User odoo dan bisa dipilih melalui Preference masing-masing User.



Gambar 204 Preference user

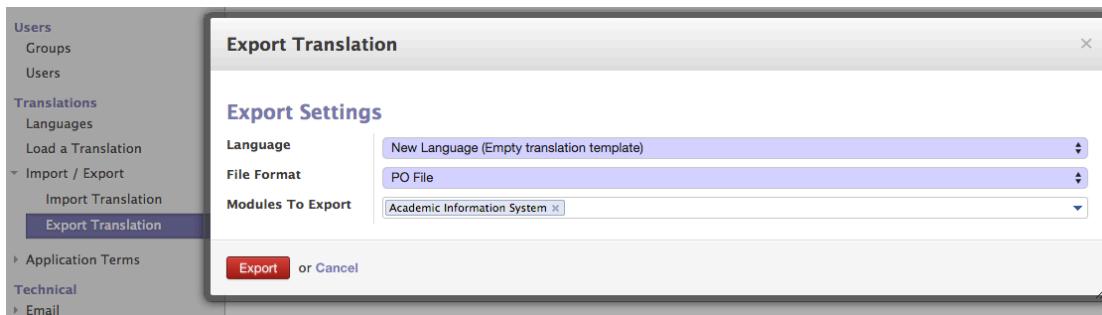
24.3 BAHASA BELUM ADA DI ODOO

24.3.1 BIKIN TEMPLATE TRANSLATE ACADEMIC.POT

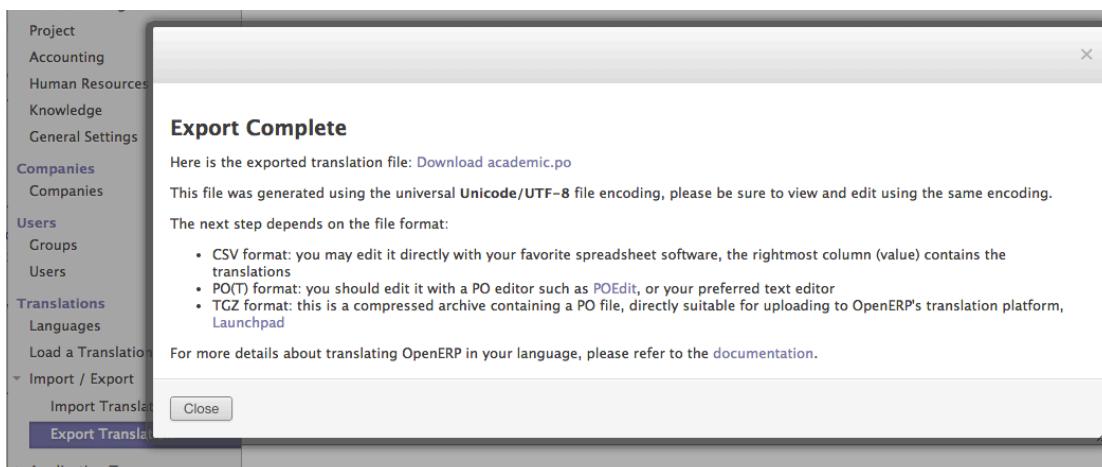
Bagian ini khusus kalau bahasa yang mau ditranslate belum ada, jadi harus bikin template-nya dulu.

Export template file translate "academic.pot" lewat menu
Settings > Translations > Import/Export >

Export Translation tanpa pilih bahasanya, terus simpan di direktori **i18n**.



Gambar 205 Export translation



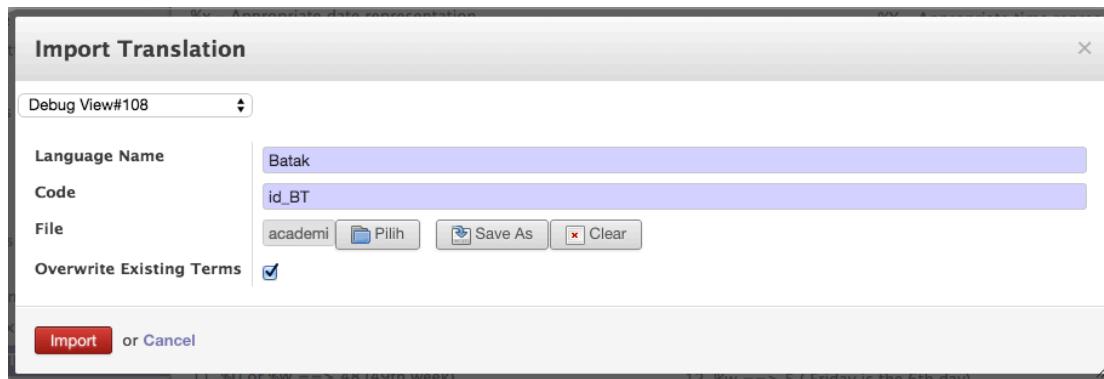
Gambar 206 Download file template academic.po

Terus rename file **academic.po** ke **academic.pot**. simpan di direktori **i18n**.

24.3.2 IMPORT KE OODOO

Kita perlu create bahasa yang baru di odoo dengan cara import file translation yang baru dibuat tadi.

Klik menu **Settings > Translations > Import/Export > Import Translation**.



Gambar 207 Import translation

Field **Language Name** adalah nama Bahasa yang akan dibuat.

Field **Code** adalah code ISO bahasa.

Field **File**, pilih file template yang barusan di download tadi, yaitu **academic.pot**.

Lanjut, bahasa yang baru dibuat muncul di daftar Languages...

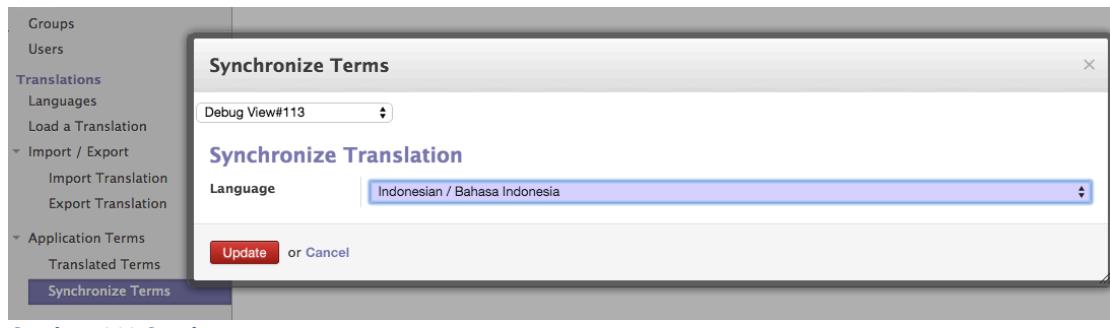
<input type="checkbox"/>	Name	Locale Code	ISO code	Direction	Translatable	Active	
<input type="checkbox"/>	English	en_US		Left-to-Right	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
<input type="checkbox"/>	English (UK)	en_GB	en_GB	Left-to-Right	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
<input type="checkbox"/>	Indonesian / Bahasa Indonesia	id_ID	id	Left-to-Right	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
<input type="checkbox"/>	Jawa	id_JW	id_JW	Left-to-Right	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
<input type="checkbox"/>	Batak	id_BT	id_BT	Left-to-Right	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

Gambar 208 Daftar bahasa yang tersedia di sistem

Lanjut ke bagian dibawah untuk proses translate.

24.4 SINKRONISASI ISTILAH

Sinkronisasi dulu istilah yang hendak di translate melalui menu **Settings > Translations > Application Terms > Synchronize Translations**.



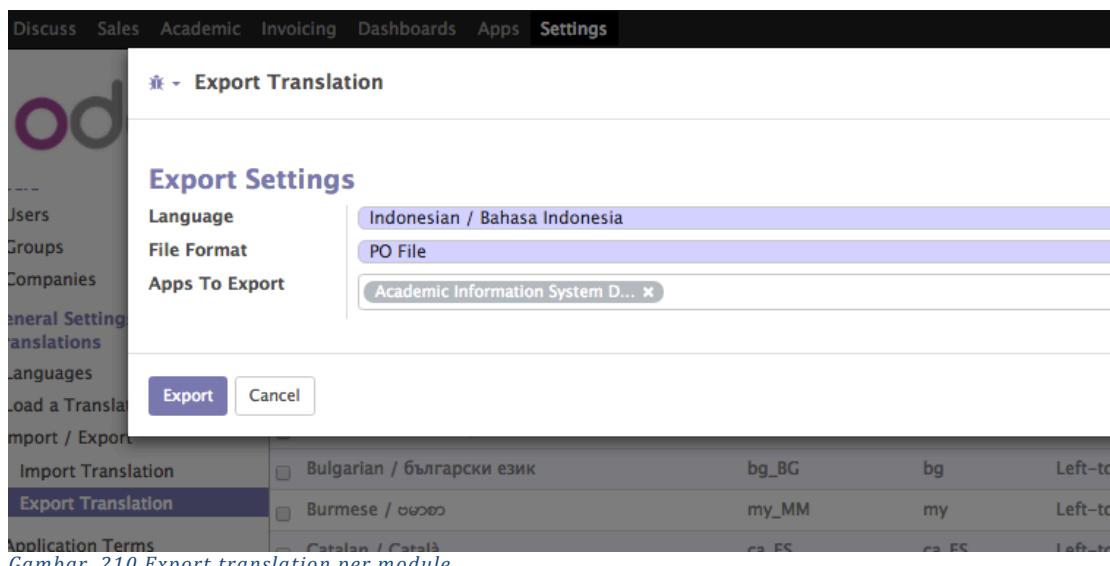
Gambar 209 Synchronize terms

24.5 BIKIN FILE TEMPLATE PER BAHASA

Bikin lagi file translate untuk Bahasa Indonesia `id.po` melalui export.

Klik menu `Settings > Translations > Import/Export > Export Translation.`

Pilih Bahasa Indonesia, simpan juga filenya di direktori `i18n`.



Gambar 210 Export translation per module

Struktur file module kita sejauh ini harus seperti berikut, ada tabahan folder `i18n` dan file-file po didalamnya...

```
!-- __init__.py
!-- __openerp__.py
```

```
|-- attendee.py
|-- attendee.xml
|-- course.py
|-- course.xml
|-- i18n
|   |-- academic.po
|   `-- id.po
|-- menu.xml
|-- partner.py
|-- partner.xml
|-- security
|   |-- group.xml
|   `-- ir.model.access.csv
|-- session.py
|-- session.xml
|-- wizard
|   |-- __init__.py
|   |-- create_attendee.py
|   `-- create_attendee_view.xml
`-- workflow.xml
```

24.6 TERJEMAHIN FILE TEMPLATE BAHASA

Buka file “`id.po`” pake `poedit` (atau sembarang text editor) dan lakukan translate istilah-istilah yang ada dengan mengganti `msgstr` setiap istilah.

Misalnya:

```
#. module: academic
#: field:academic.session,taken_seats:0
msgid "Taken Seats"
msgstr "Tempat Sudah Terpakai"

#. module: academic
#: view:academic.create.attendee.wizard:0
msgid "Add attendees"
msgstr "Tambah Peserta"
```

```
1 # Translation of OpenERP Server.
2 # This file contains the translation of the following modules:
3 #   * academic
4 #
5 msgid ""
6 msgstr ""
7 "Project-Id-Version: OpenERP Server 7.0-20130829-231103\n"
8 "Report-Msgid-Bugs-To: \n"
9 "POT-Creation-Date: 2014-09-04 09:19+0000\n"
10 "PO-Revision-Date: 2014-09-04 09:19+0000\n"
11 "Last-Translator: <>\n"
12 "Language-Team: \n"
13 "MIME-Version: 1.0\n"
14 "Content-Type: text/plain; charset=UTF-8\n"
15 "Content-Transfer-Encoding: \n"
16 "Plural-Forms: \n"
17
18#. module: academic
19#: field:academic.session,taken_seats:0
20msgid "Taken Seats"
21msgstr "Taken Seats"
22
23#. module: academic
24#: view:academic.create.attendee.wizard:0
25msgid "Add attendees"
26msgstr "Add attendees"
27
28#. module: academic
29#: selection:academic.session,state:0
30msgid "Confirmed"
31msgstr "Confirmed"
```

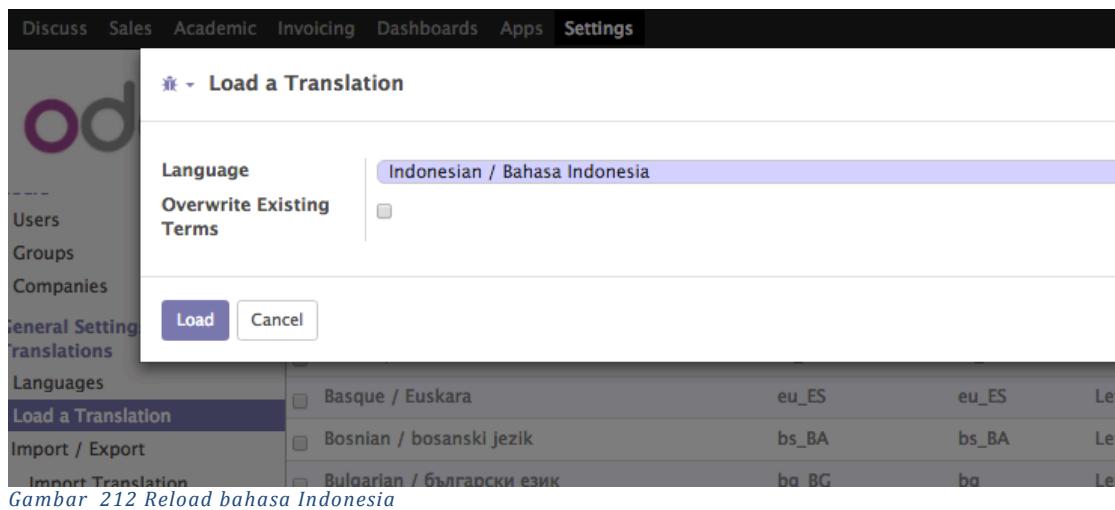
Gambar 211 File template bahasa yang harus diterjemahin

24.7 RELOAD BAHASA INDONESIA

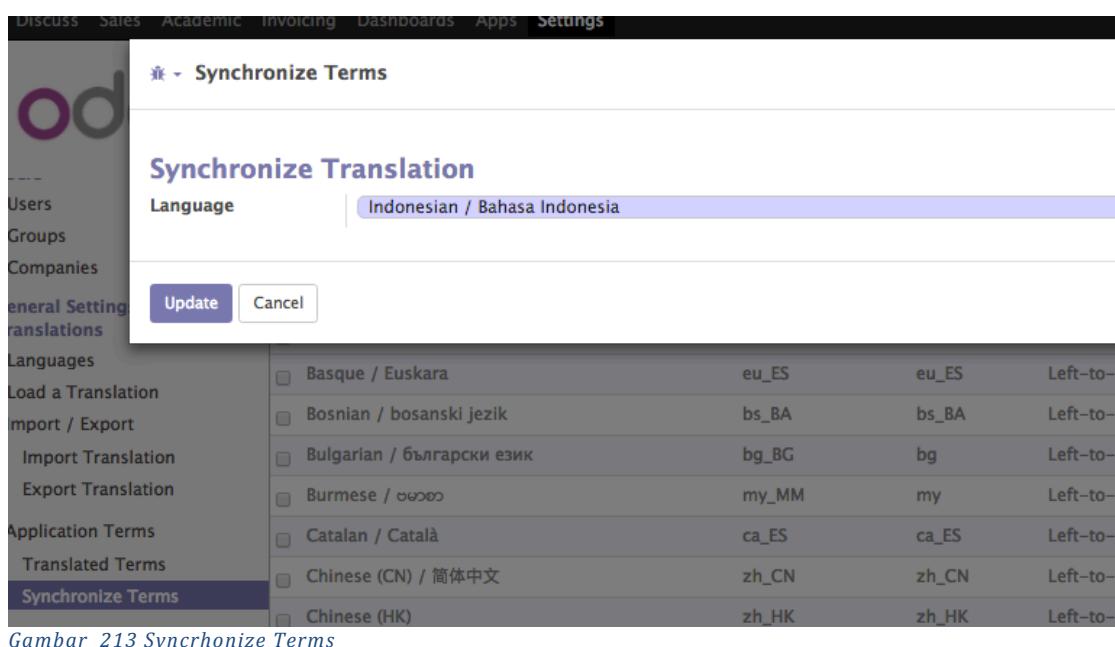
Setelah edit file terjemahan `id.po` selesai semua istilah, data ini perlu di upload ke database supaya module bisa ditampilkan dalam Bahasa Indonesia.

Caranya:

1. Upload module
2. Klik menu `Settings > Translations > Load Translation`, pilih Bahasa Indonesia



3. Klik menu **Settings > Translations > Application Terms > Syncrhonize Term**



Selesai ! Module addons academic udah bisa dipakai jika user pilih Bahasa Indonesia.

24.8 ISTILAH TAMBAHAN

Default-nya odoo POT export hanya membaca label di dalam file XML dan pada definisi field di Python code. Tapi sebetulnya semua string Python dapat dibaca asal udah diapit dengan `_()`, misalnya `_('Label')`.

Modif file `session.py`, `attendee.py`, `course.py`, tambahi
import paket berikut..

```
from tools.translate import _
```

Udah itu tambah operator "`_`" di setiap tempat yang diperlukan,
lalu ulangi langkah mulai dari Synchronize Terms di atas.

25 REPORT RML

Catatan: Sistem report RML hanya berlaku di OpenERP versi 7.0.
Untuk versi 10, report menggunakan WEB.

25.1 INSTALASI

Instal module `base_report_designer`.

Setalah instal berhasil, akan muncul popup dimana kita bisa download Report Designer Plug-in, yaitu file `odoo_report_designer.zip`.



Gambar 214 Install module `base_report_designer`

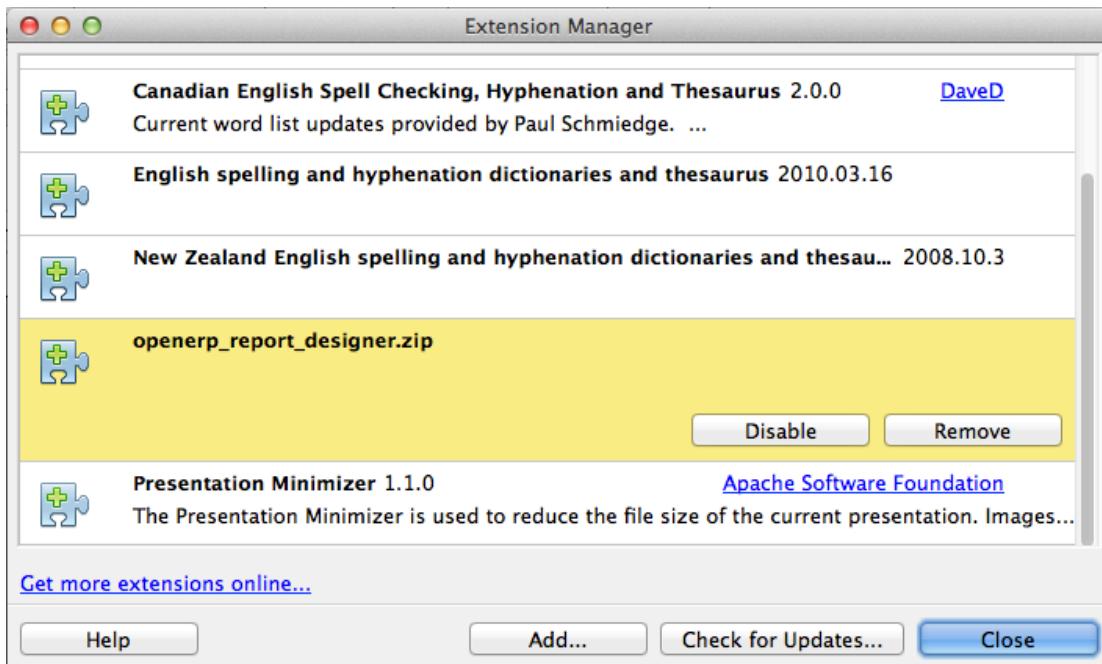
25.2 INSTALL PLUG-IN DI OPENOFFICE

Buka window Extension Manager dari Menu Bar Openoffice Writer, yaitu menu `Tools > Extension Menu`.

Klik tombol `Add`.

Pilih lokasi file `odoo_report_designer.zip` yang tadi udah didownload.

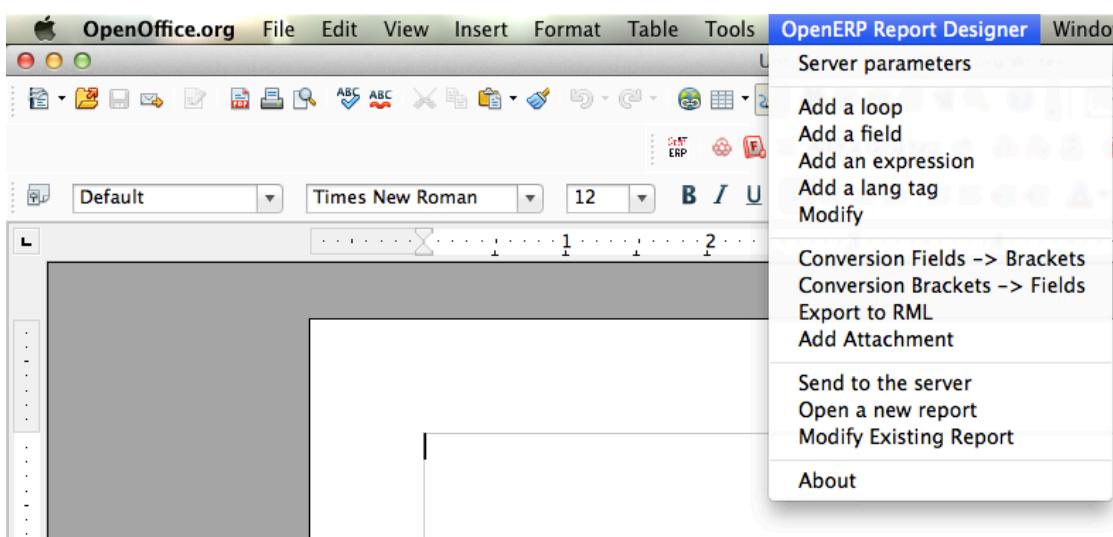
Setelah selesai nambah package kita bisa lihat di daftar 'Extension Manager' dan statusnya harus 'Enabled'.



Gambar 215 Install plug in OpenOffice

Restart OpenOffice writer.

Menu odoo Report Designer muncul di OpenOffice.



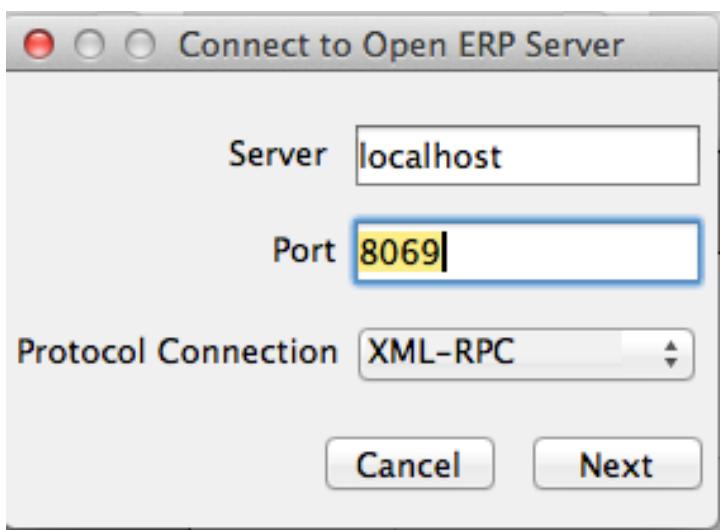
Gambar 216 Menu odoo Report Designer pada OpenOffice

25.3 KONFIGURASI

Berikut ini langkah konfigurasi plug-in odoo Report Designer pada Openoffice writer.

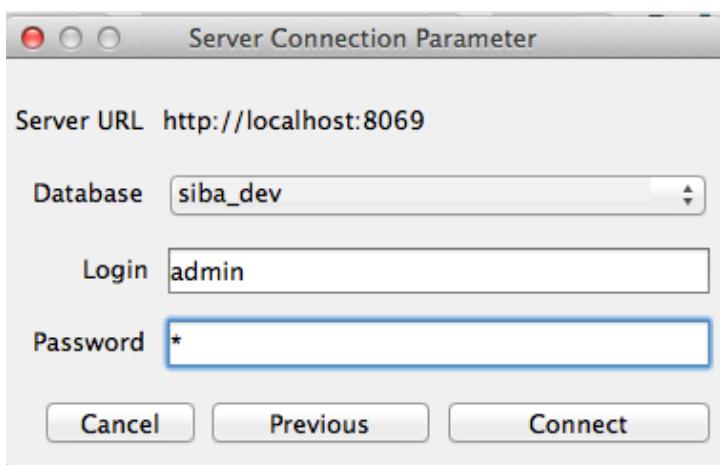
Connect ke odoo Server lewat Menu bar **odoo Report Designer > Server parameter**.

Masukkan Server URL, database, user name, dan password untuk masuk ke odoo.



Gambar 217 Koneksi ke server

Klik "Next".



Gambar 218 Login ke server

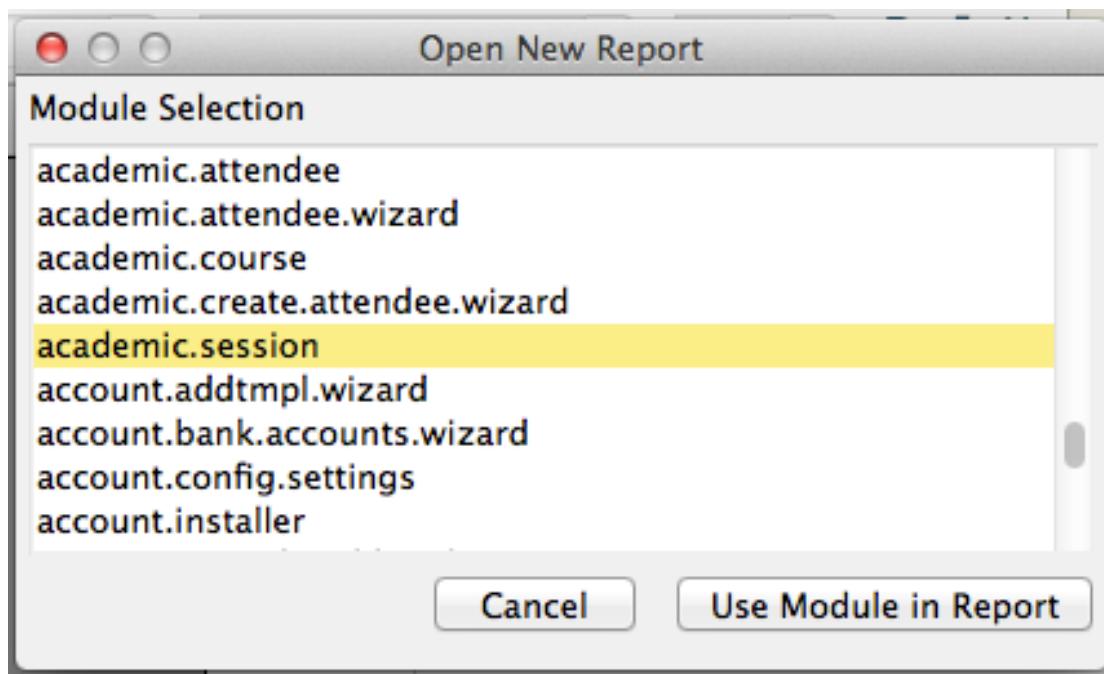
Klik "Connect". Jika berhasil akan muncul pesan 'You can start creating your report in current document.'

25.4 BIKIN REPORT BARU

Coba kita bikin report untuk Session object, yang menampilkan data Session name, date, duration, duration, responsible name serta daftar peserta yang hadir (Attendees).

Klik menu di OpenOffice: `odoo Report Designer > Open a New Report.`

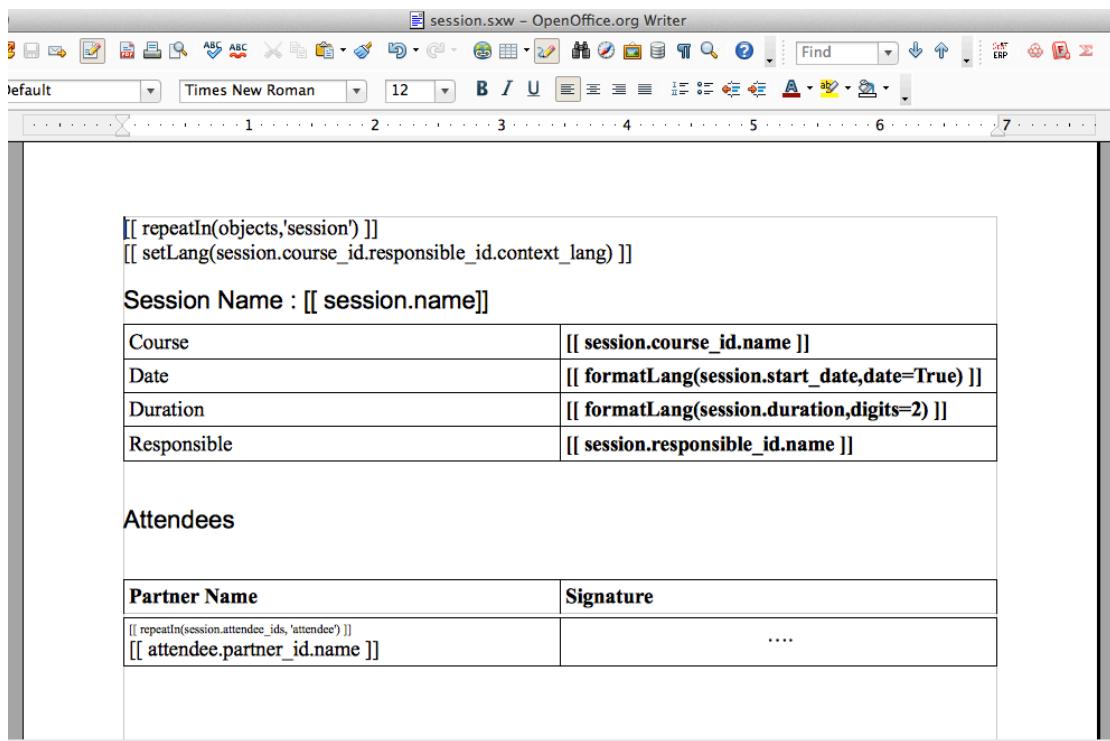
Pilih nama object yang mau dibuat reportnya, yaitu `academic.session`.



Gambar 219 Bikin report baru

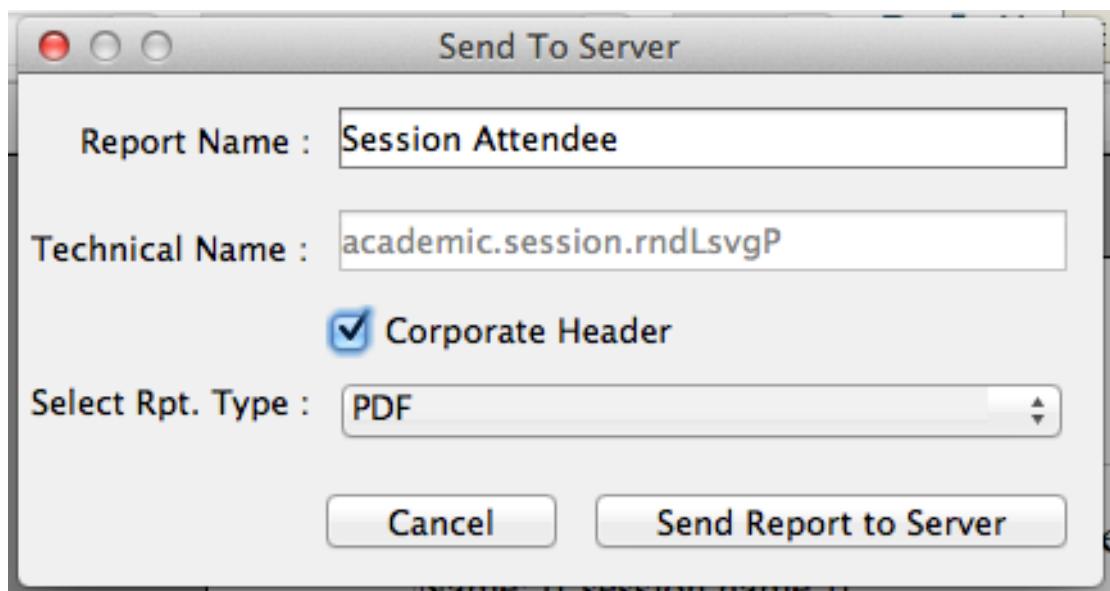
Klik `Use Module in Report`.

Bikin template seperti dibawah ini pada OpenOffice Writer.



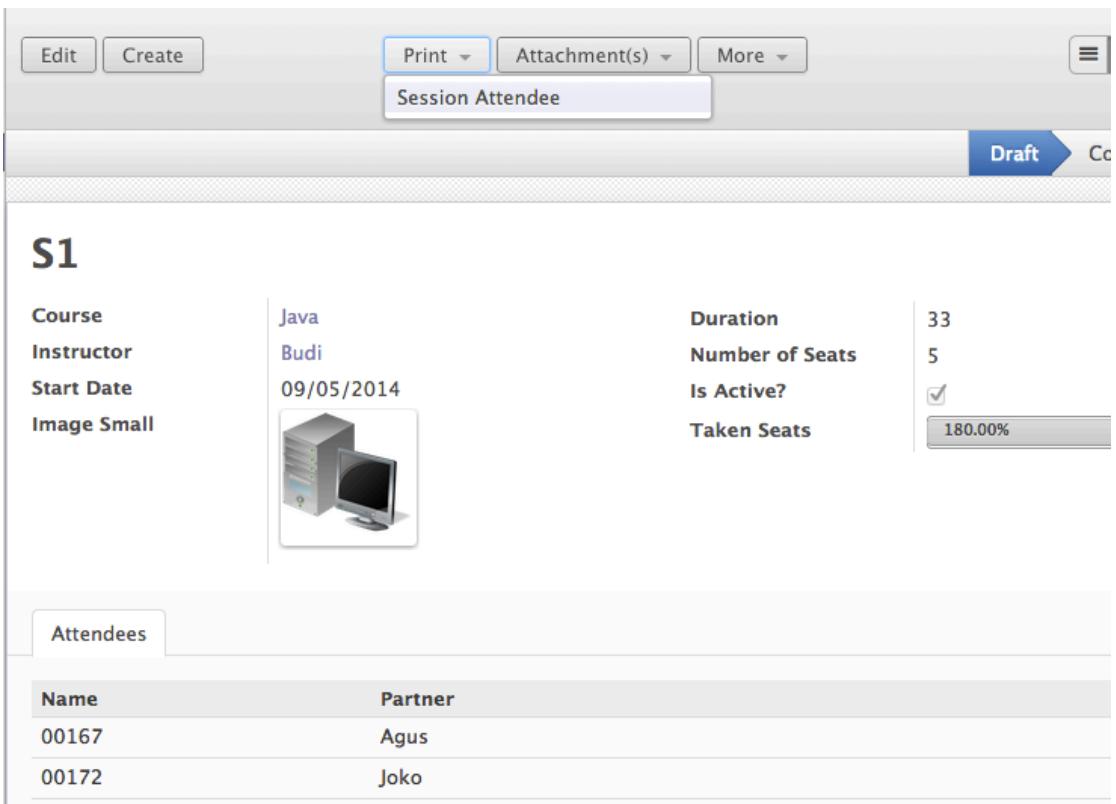
Gambar 220 Template di OpenOffice Writer

Save file, lalu klik menu **odoo Report Designer > Send Report to Server.**



Gambar 221 Kirim report ke server

Otomatis akan muncul menu Print > Session Attendee di form view Session.



The screenshot shows the Odoo interface for creating a new session. At the top, there are buttons for 'Edit', 'Create', 'Print' (which is highlighted), 'Attachment(s)', 'More', and a menu icon. Below the toolbar, the session details are listed: Course (Java), Instructor (Budi), Start Date (09/05/2014), and an image placeholder for Image Small. To the right, session parameters are shown: Duration (33), Number of Seats (5), Is Active? (checked), and Taken Seats (180.00%). At the bottom, there is a table for 'Attendees' with two entries: Name (00167, 00172) and Partner (Agus, Joko). A blue arrow at the bottom right indicates the process flow.

Gambar 222 Report langsung tersedia melalui tombol Print di form view Session

Ketika diklik akan terbentuk file PDF yang hasilnya sesuai dengan template yang udah dibuat di OpenOffice.

Phone: 12345678, 89900020
Mail: info@yourcompany.com

Session Name : S1

Course	Java
Date	09/05/2014
Duration	33.00
Responsible	

Attendees

Partner Name	Signature
Agus
Joko
Joko
Agus
Cabang A
Customer
Administrator
Administrator
Joko

Gambar 223 Report PDF

25.5 SYNTAX TEMPLATE

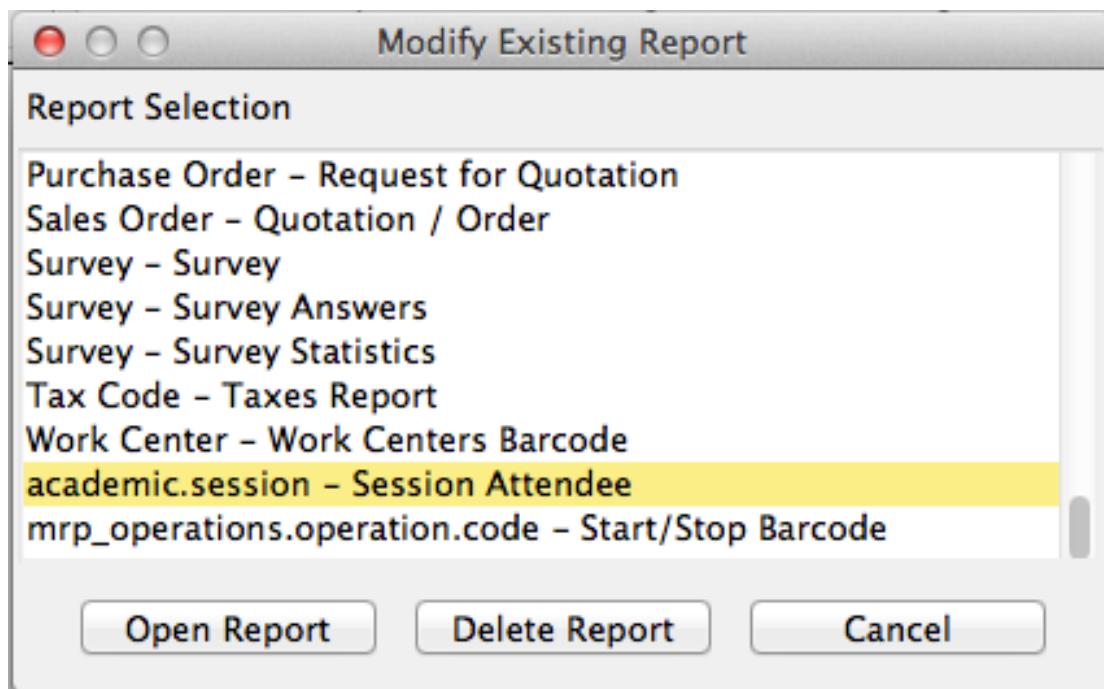
Expresi yang boleh dipakai pada report templates odoo:

<i>objects</i>	Berisi array list dari record yang mau di-print
<i>data</i>	Berasal dari wizard yang menjalankan report
<i>user</i>	User yang sedang login (as per browse())
<i>time</i>	Python time module
<i>repeatIn(list,'var','tag')</i>	Ulangi parent element dimana dia dipanggil dan bernama tag untuk setiap object yang ada pada list, dan membuat object tersebut tersedia sebagai var selama loop
<i>setTag('tag1','tag2')</i>	Ganti parent RML tag1 dengan tag2
<i>removeParentNode('tag')</i>	Hapus parent RML element bernama tag
<i>formatLang(value, digits=2, date=False, date_time=False,</i>	Untuk mem-format date, time atau nominal

grouping=True, monetary=False)	uang sesuai locale
<code>setLang('lang_code')</code>	Se bahasa dan locale untuk penterjemahan

25.6 MODIFY EXISTING REPORT

Dari menu `odoo Report Designer > Modify Existing Report`, pilih report yang akan dimodif



Gambar 224 Modif report

Akan terbuka lagi template asli report tersebut, lakukan edit seperti sebelumnya.

Jika sudah selesai, kirim balik ke server lewat menu `odoo Report Designer > Send Report to Server`.

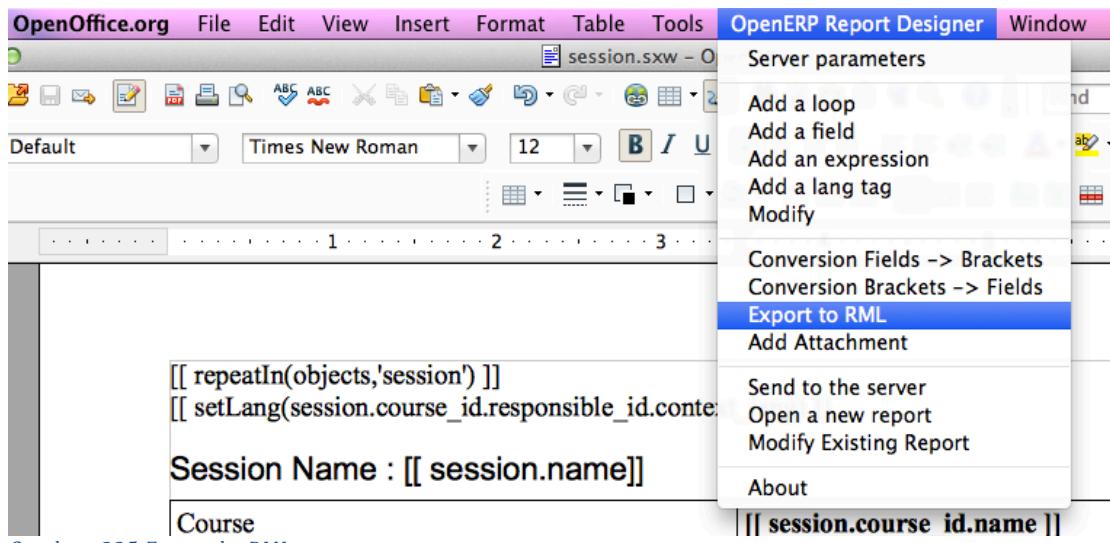
Semua report odoo yang tersedia di atas bisa diedit untuk penyesuaian sesuai kebutuhan perusahaan masing-masing.

25.7 REPORT DI ADDONS – RML

Buat folder “report” di dalam folder addons academic.

Export OpenOffice report template ke sebuah file RML dan simpan file RML di dalam report folder.

Save sebagai `session_attendee.rml`.



Gambar 225 Export ke RML

Tambah satu file XML baru untuk deklarasi report. Kasi nama misalnya `report/session_report.xml`. Isinya seperti ini.

```
1 <openerp>
2   <data>
3     <report id="session_report"
4       string="Print Session Report"
5       model="academic.session"
6       name="session.report"
7       rml="academic/report/session.rml"
8     />
9   </data>
10 </openerp>
```

Gambar 226 Bikin tag report

Edit file `__openerp__.py` dan tambahi file
`report/session_report.xml`.

```
1  {
2      "name": "Academic Information System",
3      "version": "1.0",
4      "depends": ["base", "board"],
5      "author": "Author Name",
6      "category": "Education",
7      "description": """\n8      this is my academic information system module\n9      """,\n10     "data": ["menu.xml",
11             "course.xml",
12             "session.xml",
13             "attendee.xml",
14             "partner.xml",
15             "workflow.xml",
16             "security/group.xml",
17             "security/ir.model.access.csv",
18             "report/session_report.xml",
19             ],
20     "installable": True,
21     "auto_install": False,
22 }
```

Gambar 227 Include report/session_report.xml

Restart odoo dan update module. Maka report Session akan tersedia di Session form view.

26 REPORT WEBKIT

Catatan: Sistem report Webkit berlaku di OpenERP versi 7.0 dan 8.0. Untuk versi 9/10, semua report menggunakan QWEB.

26.1 INSTALASI

Instal `report_webkit` report module.

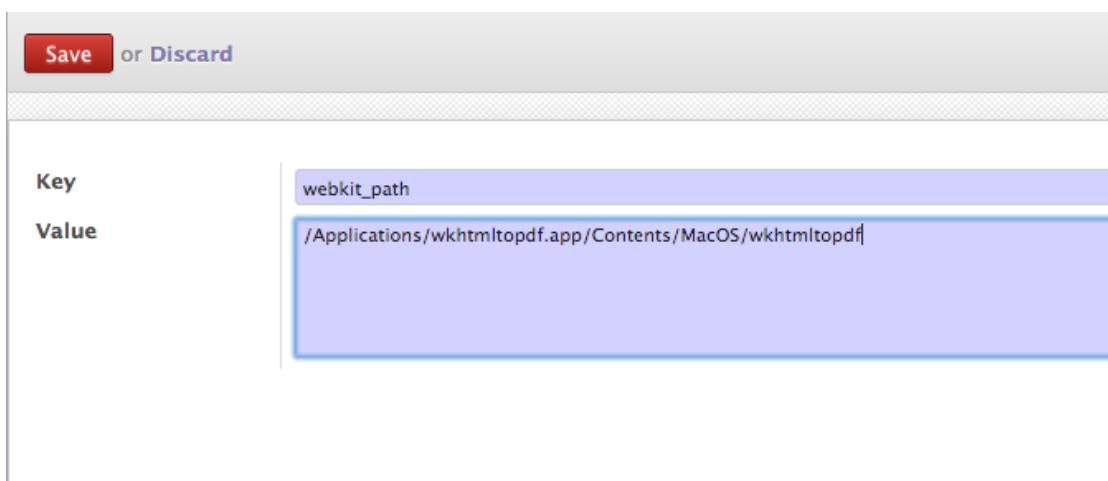
Instal program `wkhtmltopdf`.

Modul ini perlu program tambahan namanya `wkhtmltopdf` yang gunanya untuk menkonvert HTML ke PDF. Minimal version 0.9.9 download dulu dari

<http://code.google.com/p/wkhtmltopdf>. Tersedia untuk Linux, Mac OS X (i386) dan Windows (32bits).

Abis instalasi di server odoo, kita perlu menge-set PATH `wkhtmltopdf` di database yang dipakai.

Caranya lewat menu `Settings > Technical > System Parameters..`



Gambar 228 Setup system parameter `webkit_path`

Tambahkan:

Key = `webkit_path`

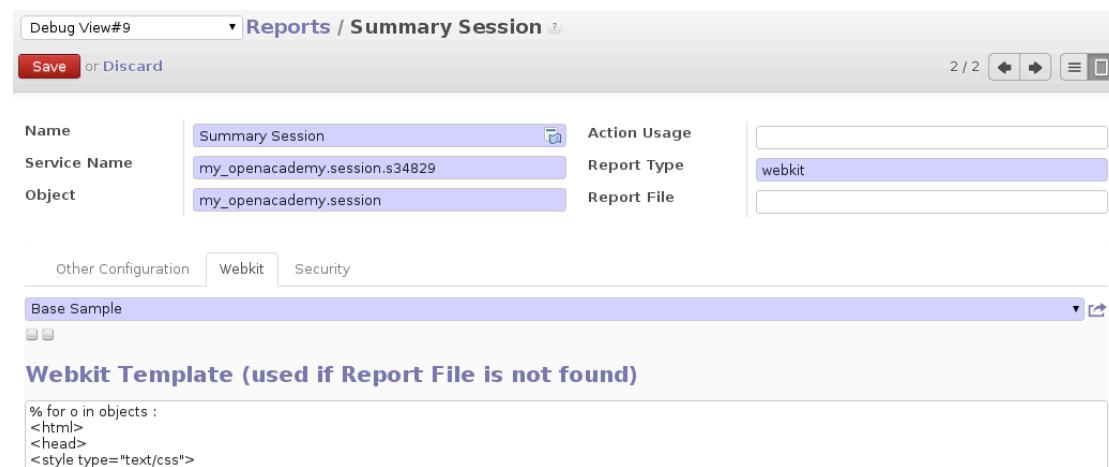
Value = lokasi program `wkhtmltopdf` di server

26.2 TEST BIKIN REPORT MANUAL

26.2.1 CREATE TEMPLATE

Klik menu **Settings > Technical > Action > Reports.**

Klik Create New .



Gambar 229 Bikin action report manual

Masukkan data sebagai berikut:

Field **Name**: the report name

Field **Service Name**: <model>.<unique number>

Field **Object**: model name

Field **Report Type**: ketik webkit

Pada **Webkit** tab:

Field **Header**: select one of the available header.

Field The Webkit Template: ketikkan report template dalam format HTML, misalnya:

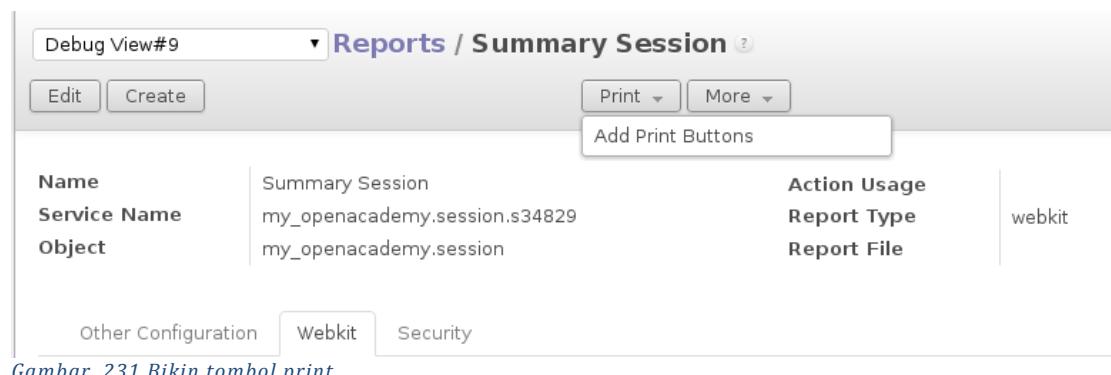
```
1 % for o in objects :
2 <html>
3 <head>
4     <style type="text/css">
5         ${css}
6         .page {page-break-after: always}
7     </style>
8 </head>
9
10 <body>
11     <div class="page">
12         <h1>${o.name}</h1>
13         <h2>Attendees</h2>
14         <table border="1" width="100%">
15             % for a in o.attendee_ids:
16             <tr><td>${a.partner_id.name}</td></tr>
17         % endfor
18     </table>
19 </div>
20 </body>
21
22 </html>
23 % endfor
24
```

Gambar 230 Template mako

26.2.2 BIKIN ACTION BUTTON

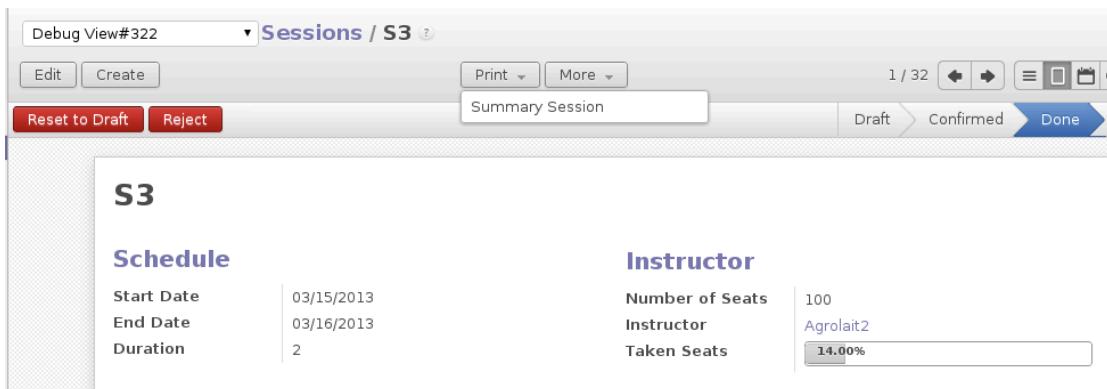
Simpan Action Report di atas.

Klik tombol Print > Add Print Buttons



Gambar 231 Bikin tombol print

Ini akan membuat record Action Binding secara otomatis sehingga tombol Print akan muncul pada form dan list view model yang bersangkutan.



Gambar 232 Tombol print muncul di form view Session

26.3 INSTALL REPORT DARI ADDONS

26.3.1 BUAT FILE TEMPLATE

Disini kita akan membuat report Webkit untuk Session object yang menampilkan session name, date, duration, durasi, responsible name dan daftar nama Attendees.

Di dalam folder `academic`, buat file `report/session.mako` isinya seperti ini.

```

1 <html>
2 <head>
3 <style type="text/css">${css}</style> </head>
4 <body>
5   <h1>Session Report</h1>
6   % for session in objects:
7     <h2>${session.course_id.name} - ${session.name}</h2>
8     <p>From ${formatLang(session.start_date, date=True)}.</p>
9     <p>Attendees:
10      <ul>
11        % for att in session.attendee_ids:
12          <li>${att.partner_id.name}</li>
13        % endfor
14      </ul>
15    </p>
16  % endfor
17 </body>
18 </html>
```

Gambar 233 Template mako

26.3.2 BUAT XML

Buat file `report/session.xml` yang berisi deklarasi record report action:

```
1 <openerp>
2     <data>
3
4         <report id="report_webkit"
5             model="academic.session"
6             name="academic.session.report"
7             file="academic/report/session.mako"
8             string="Session Report"
9             report_type="webkit"
10        />
11    </data>
12 </openerp>
13
```

Gambar 234 Bikin action report di XML

Tambahkan file XML diatas pada `__openerp__.py`:

```
1 {
2     "name": "Academic Information System",
3     "version": "1.0",
4     "depends": ["base"],
5     "author": "Author Name",
6     "category": "Education",
7     "description": """
8 this is my academic information system module
9 """,
10    "data": ["menu.xml",
11            "course.xml",
12            "session.xml",
13            "attendee.xml",
14            "partner.xml",
15            "workflow.xml",
16            "security/group.xml",
17            "security/ir.model.access.csv",
18            "wizard/create_attendee_view.xml",
19            "report/session.xml"
20        ],
21    "installable": True,
22    "auto_install": False,
23 }
```

Gambar 235 Panggil dari `__openerp__.py`

Restart odoo dan update module, hasilnya:

Screenshot of a software interface showing session details and attendee information.

Session Details:

Course	Java	Duration	33
Instructor	Budi	Number of Seats	5
Start Date	09/05/2014	Is Active?	<input checked="" type="checkbox"/>
Image Small		Taken Seats	180.00%

Attendees:

Name	Partner
00167	Agus
00172	Joko

Gambar 236 Muncul tombol print report webkit

Muncul menu report baru di bawah tombol Print berupa report webkit. Klik menu report akan muncul report dalam bentuk PDF.

Phone: 12345678, 89900020
 Mail: info@yourcompany.com

Session Report

Java - S1

From 09/05/2014.

Attendees:

- Agus
- Joko
- Joko
- Agus
- Cabang A
- Customer
- Administrator
- Administrator
- Joko

Gambar 237 Report PDF

27 REPORT QWEB

Sejak versi 9 dan 10, semua reporting Odoo menggunakan QWEB.

Untuk membuat report QWEB terhadap semua object Odoo, lakukan langkah-langkah berikut...

27.1 BUAT FOLDER REPORT

Semua file report akan ditempatkan di dalam folder ini supaya nggak tercampur dengan file-file XML lainnya..

27.2 BUAT FILE XML REPORT UNTUK MENU REPORT

Next, buat file XML report khusus per object, contohnya untuk object session, dikasi nama session.xml.

Di dalam file ini, ada beberapa record yang perlu dicantumin, pertama...

```
<?xml version="1.0" encoding="utf-8"?>
<openerp>
    <data>

        <report
            id="report_session_menu"
            string="Session"
            model="academic.session"
            report_type="qweb-pdf"
            file="academic.session"
            name="academic.session_report"
        />
```

ini adalah definisi record untuk menu action report pada object academic.session (field model).

Label menu tersebut adalah Session (field string)...

Type report adalah qweb-pdf, dan file PDF nya nanti bernama academic.session (field report_type dan file)..

Terakhir, field name, menentukan nama template QWEB yang akan dipanggil pada saat menu action dipanggil, yaitu

`academic.session_report...` Disini academic adalah nama folder addons kita.

27.3 XML RECORD UNTUK TEMPLATE QWEB

Lanjut, buat record XML untuk definisi template yang dipanggil di menu action diatas, yaitu yang ID nya `academic.session_report...`

```
<template id="session_report">
    <t t-call="report.html_container">
        <t t-foreach="docs" t-as="doc">
            <t t-call="academic.session_report_document"/>
        </t>
    </t>
</template>
```

Ini template isinya cuma manggil template lain lagi yaitu `report.html_container` untuk memanggil header dan footer standard report Odoo.

Lalu dilakukan looping foreach untuk setiap variabel `docs`. Variable ini adalah record list dari Session yang hendak diprint, baik melalui form view maupun list view dengan cara men-select 1 atau lebih Session yang mau diprint.

Pada setiap looping variabel `docs`, kita simpan ke local variabel `doc`, lalu kita panggil lagi template lain yang namanya `academic.session_report_document`.

Pada template terakhir inilah layout report per satu record session kita definisikan bentuknya seperti apa.

27.4 XML RECORD UNTUK REPORT PER RECORD SESSION

Lanjut, ini definisi layout report per setiap satu record session yang hendak diprint.

```
<template id="session_report_document">
    <t t-call="report.external_layout">
        <t t-set="doc" t-value="doc.with_context(
```

```

{'lang':doc.instructor_id.lang})" />
<div class="page">

    <div class="oe_structure"/>

    <h2>
        SESSION: <span t-field="doc.name"/>
    </h2>

    <table class="table table-condensed">
        <tr>
            <td>Course</td>
            <td><span t-field="doc.course_id"/></td>
            <td>Instructor</td>
            <td><span t-field="doc.instructor_id"/></td>
        </tr>
        <tr>
            <td>Start Date</td>
            <td><span t-field="doc.start_date"/></td>
            <td>Duration</td>
            <td><span t-field="doc.duration"/></td>
        </tr>
        <tr>
            <td>Taken Seats</td>
            <td><span t-field="doc.taken_seats"/></td>
            <td>Active</td>
            <td><span t-field="doc.active"/></td>
        </tr>
    </table>

    <table class="table table-condensed">
        <thead>
            <tr>
                <th>No</th>
                <th>Partner</th>
                <th>Signed</th>
            </tr>
        </thead>

        <tbody>
            <tr t-foreach="doc.attendee_ids" t-as="l">
                <td>
                    <span t-field="l.name"/>
                </td>
                <td>
                    <span t-field="l.partner_id" />
                </td>
                <td>
                </td>
            </tr>
        </tbody>
    </table>

    <div class="oe_structure"/>
</div>
</t>
</template>

```

Inti dari template di atas adalah...

Bahwasanya template ini mendapat variabel doc yang isinya record session yang hendak diprint, dari object ini kita bisa panggil semua field yang ada pada session.

Pertama-tama kita panggil template external_layout bawaan Odoo untuk mengatur layout standard report Odoo..

```
<template id="session_report_document">
  <t t-call="report.external_layout">
```

Lalu kita set bahasa yang akan digunakan pada report sesuai dengan bahasanya Instruktur...

```
<t t-set="doc" t-value="doc.with_context(
    {'lang':doc.instructor_id.lang})" />
```

Kemudian kita gunakan class CSS page untuk ukuran kertas, keluarkan tag HTML H2 untuk menampilkan judul report...

```
<div class="page">
  <div class="oe_structure"/>
  <h2> SESSION: <span t-field="doc.name"/>
</h2>
```

Kemudian.. kita buat table untuk menampilkan data header session...

```
<table class="table table-condensed">
  <tr>
    <td>Course</td>
    <td><span t-field="doc.course_id"/></td>
    <td>Instructor</td>
    <td><span t-field="doc.instructor_id"/></td>
  </tr>
  <tr>
    <td>Start Date</td>
    <td><span t-field="doc.start_date"/></td>
    <td>Duration</td>
    <td><span t-field="doc.duration"/></td>
  </tr>
  <tr>
    <td>Taken Seats</td>
    <td><span t-field="doc.taken_seats"/></td>
    <td>Active</td>
```

```
</td><span t-field="doc.active"/></td>
</tr>
</table>
```

kita lihat cara menampilkan isi field record session, yaitu dengan notasi doc.<fieldname> misalnya doc.name, doc.instructor_id, dst...

Lalu kita tampilkan detail attendee dalam bentuk tabel lagi...

```
<table class="table table-condensed">
  <thead>
    <tr>
      <th>No</th>
      <th>Partner</th>
      <th>Signed</th>
    </tr>
  </thead>

  <tbody>
    <tr t-foreach="doc.attendee_ids" t-as="l">
      <td>
        <span t-field="l.name"/>
      </td>

      <td>
        <span t-field="l.partner_id" />
      </td>
      <td>
        </td>
    </tr>
  </tbody>
</table>
```

disini diliat kita melakukan looping for each untuk setiap attendee_ids milik session yang diprint.

Lalu pada setiap record attendee_id yang disimpan pada variabel l, kita keluarkan lagi isi fieldnya dengan cara l.name dan l.partner_id...

Restart odoo dan update module ...

Hasilnya, muncul menu action Print di list view Session ketika ada yang di-select...

Daftar Session									Search...	
			Print		Action	Filters		Group By		1-2 / 2
			Session		Favorites					< >
Course	Name	Instructor	Start Date		Duration		Seats	Active	Taken Seat	
Java	sesion1	Joko	02/22/2017		20		100	<input checked="" type="checkbox"/>		
Java	Session3	Mukidi	02/22/2017		10		10	<input checked="" type="checkbox"/>		

Muncul juga menu action print di setiap form view session...

Daftar Session / sesion1

		Print		Action					Draft	Confirmed
		Session								
sesion1										
Course	Java	Duration	20	Seats	100	Active	<input checked="" type="checkbox"/>	Taken Seat	3%	
Instructor	Joko									
Start Date	02/22/2017									
Image Small										
Attendees										
Name		Partner								
01		Agus								
02		Badu								
03		Budi								

Ketika di click, maka akan dihasilkan report PDF session sesuai dengan layout yang sudah ditentukan sebelumnya...

The screenshot shows a Mac OS X desktop with a PDF viewer window open. The window title is "Session (4).pdf (page 1 of 2)". The PDF content includes:

- A form titled "SESSION Master" with fields: No, Partner, Status, Seats, Duration, and Signed. A blue "1" is overlaid on the bottom right corner.
- A form titled "SESSION Session1" with fields: No, Partner, Status, Seats, Duration, and Signed. A blue "2" is overlaid on the bottom right corner.
- A header section with the "odoo" logo and "My Company Tagline".
- A section titled "SESSION: sesion1" with the following table:

Course	Java	Instructor	Joko
Start Date	02/22/2017	Duration	20
Taken Seats	3.0	Active	True
- A table with columns "No", "Partner", and "Signed":

No	Partner	Signed
01	Agus	
02	Badu	
03	Budi	

Selesai ... kalau nggak error 😊

28 DASHBOARD

Disini kita akan membuat dashboard yang berisi graph view yang udah kita buat sebelumnya pada bagian Advanced View, sessions calendar view dan daftar courses (dan bisa dipindahkan ke form view).

Kita akan buat file XML baru untuk dashboard di dalam folder addons. Isinya adalah:

- the board view,
- the actions referenced in that view,
- an action to open the dashboard,
- as well as a re-definition of the main menu item “academic_menu” to add the action to open the dashboard

28.1 BIKIN SUB MENU DASHBOARD

Create file `dashboard.xml`. Isinya adalah

```
<openerp>
  <data>

    <!-- tambah menu level 1 Dashboard -->
    <menuitem id="menu_dashboard"
      name="Dashboard" sequence="0"
      parent="academic_top" />

    <!-- tambah menu sub level 2 -->
    <menuitem
      name="Session Dashboard"
      parent="menu_dashboard"
      action="open_board_session"
      sequence="1"
      id="menu_board_session"
      icon="terp-graph"/>
  </data>
</openerp>
```

Gambar 238 Menu dashboard

Update file `__openerp__.py` dan tambahkan file `dashboard.xml` yang baru dibuat tadi.

Jangan lupa tambahi depends ke modul "board" supaya module dashboard dikenal dari modul academic.

```
{  
    "name": "Academic Information System Day 5",  
    "version": "1.0",  
    "depends": [  
        "base",  
        "account",  
        "sale",  
        "board",  
    ],  
    "author": "akhmad.daniel@gmail.com",  
    "category": "Education",  
    'website': 'http://www.vitraining.com',  
    "description": """\nAcademic Information System Day 5  
-----  
* Add report QWEB  
* Add dashboard  
  
"""",  
    "data": [  
        "menu.xml",  
        "course.xml",  
        "session.xml",  
        "attendee.xml",  
        "partner.xml",  
        "workflow.xml",  
        "security/group.xml",  
        "security/ir.model.access.csv",  
        "wizard/create_attendee.xml",  
        "report/session.xml",  
        "dashboard.xml",  
    ],  
    "installable": True,  
    "auto_install": False,  
    "application": True,  
}  
Gambar 239 Depends board module dan panggil dashboard.xml dari __openerp__.py
```

28.2 TAMBAHI DASHBOARD XML

Edit `dashboard.xml` tambahi board sebagai berikut:

```
<record model="ir.ui.view" id="board_session_form">  
    <field name="name">Session Dashboard Form</field>  
    <field name="model">board.board</field>  
    <field name="type">form</field>
```

```

<field name="arch" type="xml">
    <form string="Session Dashboard" version="7.0">
        <board style="2-1">
            <column>
                <action
                    string="Attendees by course"
                    name="%{act_session_graph}d"
                    colspan="4"
                    height="150"
                    width="510" />
                <action
                    string="Sessions"
                    name="%{act_session_calendar}d"
                    colspan="4" />
                </column>
                <column>
                    <action
                        string="Courses"
                        name="%{act_course_list}d"
                        colspan="4" />
                </column>
            </board>
        </form>
    </field>
</record>

```

Gambar 240 Definisi dashboard

Tambahi definisi action yang perlu dipanggil lewat dashboard
 (contoh di atas adalah `act_session_calendar`,
`act_session_graph`, dan `act_course_list`).

Deklarasinya harus di atas board session form.

```

<record model="ir.actions.act_window" id="act_session_calendar">
    <field name="name">academic.session.cal</field>
    <field name="res_model">academic.session</field>
    <field name="view_type">form</field>
    <field name="view_mode">calendar</field>
    <field name="view_id" ref="session_cal"/>
</record>

<record model="ir.actions.act_window" id="act_session_graph">
    <field name="name">academic.session.graph</field>
    <field name="res_model">academic.session</field>
    <field name="view_type">form</field>
    <field name="view_mode">graph</field>
    <field name="view_id" ref="session_graph"/>
</record>

<record model="ir.actions.act_window" id="act_course_list">
    <field name="name">academic.course.list</field>
    <field name="res_model">academic.course</field>
    <field name="view_type">form</field>
    <field name="view_mode">tree,form</field>
</record>

```

Gambar 241 Action window untuk content dashboard

Tambahi action window yang dipanggil oleh menuitem
(deklarasinya harus di atas menu item)

```
<record model="ir.actions.act_window" id="open_board_session">
    <field name="name">Session Dashboard</field>
    <field name="res_model">board.board</field>
    <field name="view_type">form</field>
    <field name="view_mode">form</field>
    <field name="usage">menu</field>
    <field name="view_id" ref="board_session_form"/>
</record>
```

Gambar 242 Action window dashboard

Selengkapnya file dashboard.xml seperti ini...

```
<openerp>
    <data>

        <record model="ir.actions.act_window" id="act_session_calendar">
            <field name="name">academic.session.cal</field>
            <field name="res_model">academic.session</field>
            <field name="view_type">form</field>
            <field name="view_mode">calendar</field>
            <field name="view_id" ref="session_cal"/>
        </record>

        <record model="ir.actions.act_window" id="act_session_graph">
            <field name="name">academic.session.graph</field>
            <field name="res_model">academic.session</field>
            <field name="view_type">form</field>
            <field name="view_mode">graph</field>
            <field name="view_id" ref="session_graph"/>
        </record>

        <record model="ir.actions.act_window" id="act_course_list">
            <field name="name">academic.course.list</field>
            <field name="res_model">academic.course</field>
            <field name="view_type">form</field>
            <field name="view_mode">tree,form</field>
        </record>

        <record model="ir.ui.view" id="board_session_form">
            <field name="name">Session Dashboard Form</field>
            <field name="model">board.board</field>
            <field name="type">form</field>
            <field name="arch" type="xml">
                <form string="Session Dashboard" version="7.0">
                    <board style="2-1">
                        <column>
                            <action
                                string="Attendees by course"
                                name="%(act_session_graph)d"
                                colspan="4"
                                height="150"
                                width="510" />
                            <action
                                string="Sessions"
                                name="%(act_session_calendar)d"
                                colspan="4" />
                        </column>
                    </board>
                </form>
            </field>
        </record>
    </data>
</openerp>
```

```

        </column>
        <column>
        <action
            string="Courses"
            name="%{act_course_list)d"
            colspan="4" />
        </column>
    </board>
</form>
</field>
</record>

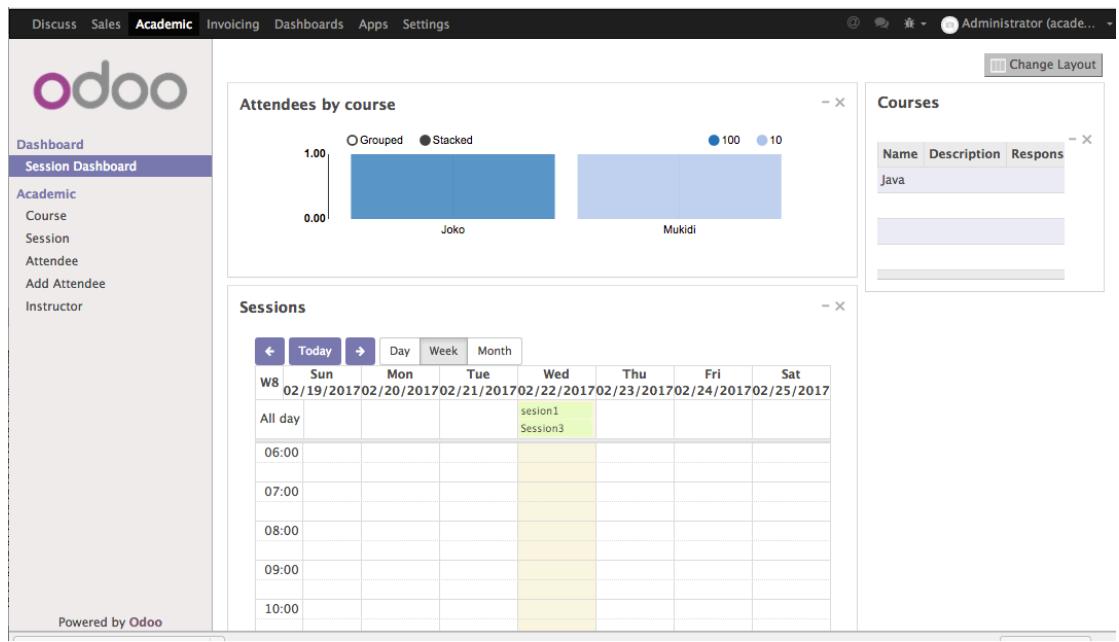
<record model="ir.actions.act_window" id="open_board_session">
    <field name="name">Session Dashboard</field>
    <field name="res_model">board.board</field>
    <field name="view_type">form</field>
    <field name="view_mode">form</field>
    <field name="usage">menu</field>
    <field name="view_id" ref="board_session_form"/>
</record>

<!-- tambah menu level 1 Dashboard -->
<menuitem id="menu_dashboard"
    name="Dashboard" sequence="0"
    parent="academic_0" />

<!-- tambah menu sub level 2 -->
<menuitem
    name="Session Dashboard"
    parent="menu_dashboard"
    action="open_board_session"
    sequence="1"
    id="menu_board_session"
    icon="terp-graph"/>
</data>
</openerp>

```

Restart odoo dan update module. Hasilnya



Gambar 243 Tampilan dashboards

29 WEB SERVICES

odoo dapat diakses melalui interface XML-RPC, dimana hampir semua bahasa pemrograman punya library-nya.

Disini kita pakai contoh kasus interfacing odoo dengan PHP.

29.1 INSTALASI XML-RPC FOR PHP

Untuk mempermudah, kita menggunakan library XML-RPC framework for PHP.

Download the XML-RPC framework for PHP dari

<http://phpxmlrpc.sourceforge.net/>

Extract file `xmlrpc-2.2.tar.gz` ke suatu folder.

29.2 AKTIFKAN PHP CURL MODULE

PHP Curl module harus aktif agar library PHP XML-RPC bisa jalan. Pastikan modulnya udah terinstal dan di-set di `php.ini`

29.3 SETUP FOLDER APLIKASI

Buat folder baru dibawah addons `academic`, kasi nama misalnya `php-xmlrpc`. Sebetulnya boleh bebas dimana aja tapi untuk memudahkan kita gabung dibawah folfer addons aja.

Ambil file `xmlrpc.inc` dari directory `lib` hasil extract XML-RPC, dan tempatkan dibawah folder `php-xmlrpc`.

Lalu buat file untuk percobaan interfacing dimana kita akan menggunakan fitur-fitur XML-RPC odoo untuk login, read, create, update, dan delete data.

Buat file PHP dengan nama `test.php` dibawah folder `php-xmlrpc`.

Isinya pada awalnya sederhana aja yaitu import library `xmlrpc.lib` dan definisi class `MyodooLib` dengan beberapa atribut untuk keperluan koneksi ke odoo:

```
1 <?php
2 include("xmlrpc.inc");
3
4 class MyOpenERPLib
5 {
6     public $user      = "admin";
7     public $password  = "1";
8     public $dbname    = "devel";
9     public $server_url= "http://127.0.0.1:8069/xmlrpc/";
10    public $id        = null;
11
12 }
13
14 ?>
15
```

Gambar 244 Bikin class PHP MyodooLib

Variable `$user` adalah nama user yang bisa melakukan login ke odoo

Variable `$password` adalah password dari user tersebut.

Variable `$dbname` adalah nama database yang akan di connect.

Variable `$server_url` adalah alamat URL server odoo.

Variable `$id` adalah integer user id yang berhasil login.

Struktur folder addons kita sejauh ini...

```
academic
|-- __init__.py
|-- __openerp__.py
|-- attendee.py
|-- attendee.xml
|-- course.py
|-- course.xml
|-- dashboard.xml
|-- menu.xml
|-- partner.py
|-- partner.xml
|-- php-xmlrpc
|   |-- test.php
|   `-- xmlrpc.inc
|-- security
|   |-- group.xml
```

```
|   '-- ir.model.access.csv
|-- session.py
|-- session.xml
|-- wizard
|   |-- __init__.py
|   '-- create_attendee.py
`-- workflow.xml
```

29.4 LOGIN

Lanjut, kita buat method `login()` di dalam class `Myodoolib`.

Deklarasi method nya sebagai berikut.

```
11
12     function login(){
13         $conn      = new xmlrpc_client($this->server_url . 'common');
14         $msg       = new xmlrpcmsg('login');
15         $msg->addParam( new xmlrpcval($this->dbname,    "string" ) );
16         $msg->addParam( new xmlrpcval($this->user,      "string" ) );
17         $msg->addParam( new xmlrpcval($this->password, "string" ) );
18         $resp      = $conn->send( $msg );
19         $val       = $resp->value();
20         $this->id    = $val->scalarval();
21         return $this->id>0 ? $this->id : -1 ;
22     }
23 }
```

Gambar 245 Method `login()`

Disini kita lakukan request login melalui XML-RPC ke odoo.

Prosesnya adalah dengan membentuk dulu object `$msg` yang merupakan class XML RPC message, `xmlrpcmsg('login')`.

Lalu kita tambahi parameter `$msg` dengan `$this->dbname`, `$this->user`, dan `$this->password`, yang terlebih dulu dibentuk menjadi object XML RPC value, `xmlrpcval`, type-nya string.

Setelah XML RPC message jadi, kita kirimkan melalui `$conn` yang adalah object XML RPC Client, `xmlrpc_client`, yang diarahkan kepada URL `$this->server_url . 'common'`.

Respon yang didapat dari server ditangkap di variable `$resp` yang masih berbentuk object. Untuk mendapatkan nilai scalarnya, perlu di convert menggunakan method `value()` dan `scalarval()`.

Hasil dari method login adalah integer berupa user id yang berhasil login atau integer -1 jika gagal login.

Test login dengan script ini:

```
193 $openerp = new MyOpenERPLib;  
194 $userId = $openerp->login();  
195 if ($userId == -1) {  
196     die("Login error ....");  
197 }  
198  
199  
200
```

Gambar 246 Test method login()

Jika login berhasil maka variable `$this->id` akan terisi dengan user id yang berhasil login dan dapat digunakan untuk keperluan method selanjutnya.

Jika gagal login, maka `$this->id` isinya -1, dan langsung exit dengan pesan "Login error....".

29.5 SEARCH

Bikin lagi method search() pada class MyodooLib. Deklarasinya seperti ini...

```

24
25     function search($relation, $key){
26         $key_array = array(
27             new xmlrpcval(
28                 array(
29                     new xmlrpcval("name", "string"),
30                     new xmlrpcval("ilike", "string"),
31                     new xmlrpcval($key, "string"),
32                 ),
33                 "array"
34             )
35         );
36
37         $msg      = new xmlrpcmsg('execute');
38         $msg->addParam(new xmlrpcval($this->dbname,    "string"));
39         $msg->addParam(new xmlrpcval($this->id,        "int"));
40         $msg->addParam(new xmlrpcval($this->password, "string"));
41         $msg->addParam(new xmlrpcval($relation,   "string"));
42         $msg->addParam(new xmlrpcval("search",     "string"));
43         $msg->addParam(new xmlrpcval($key_array,  "array"));
44         $conn     = new xmlrpc_client($this->server_url . 'object');
45         $conn->return_type = 'phpvals';
46         $resp = $conn->send($msg);
47         if ($resp->faultCode())
48         {
49             var_dump($resp);
50             return -2;
51         }
52         else {
53             if($val = $resp->value()){
54                 return $val;
55             }
56             else
57             {
58                 return -1;
59             }
60         }
61     }

```

Gambar 247 Method search()

Disini kita lakukan request search data dengan membentuk dulu object `$msg` yang merupakan class XML RPC message, `xmlrpcmsg('execute')`. Maksudnya adalah kita akan men-execute sebuah function pada suatu object.

Lalu kita tambahi parameter `$msg` dengan `$this->dbname`, `$this->user`, dan `$this->password`, yang terlebih dulu dibentuk menjadi object XML RPC value, `xmlrpcval`, type-nya string.

Parameter selanjutnya adalah nama relasi object yang akan di-search. Dalam hal ini kita jadikan input parameter `$relation`.

Parameter berikutnya adalah nama function yang akan di-execute yaitu "search".

Parameter berikutnya adalah array XML-RPC yang berisi kriteria pencarian, disini kita set pada variable \$key_array.

Pembentukan variable \$key_array dilakukan sebelumnya, yaitu merupakan array satu elemen berisi object xmlrpcval berjenis "array" dengan nilai array berisi "name", "ilike", \$key yang dicari dengan sebelumnya masing-masing dibentuk menjadi object xmlrpcval berjenis string. Berikut ini cuplikannya.

```
$key_array = array(
    new xmlrpcval(
        array(
            new xmlrpcval("name", "string"),
            new xmlrpcval("ilike", "string"),
            new xmlrpcval($key, "string"),
        ),
        "array"
    )
);
```

Test method search dengan menjalankan script berikut :

```
193
194 $openerp = new MyOpenERPLib;
195 $userId = $openerp->login();
196 if ($userId == -1) {
197     die("Login error ....");
198 }
199
200
201
202 echo "=====\\Searching data containing java.....\n";
203 $session_ids    = $openerp->search("academic.session","java");
204 var_dump($session_ids);
205
```

Gambar 248 Test method search()

Pada test di atas kita menjalankan method search() untuk mencari session yang memiliki nama ada "java"nya. Hasilnya adalah berupa array list of id dari session yang matching dengan pencarian.

29.6 READ

Lanjut, bikin method search() pada class MyodooLib.

Deklarasinya seperti ini...

```
68     function read($relation , $ids, $fields){  
69  
70         $id_val=array();  
71         foreach ($ids as $i) {  
72             $id_val[] = new xmlrpcval($i , "int");  
73         }  
74  
75         $fields_val=array();  
76         foreach ($fields as $f) {  
77             $fields_val[] = new xmlrpcval($f, "string");  
78         }  
79  
80         $msg = new xmlrpcmsg('execute');  
81         $msg->addParam(new xmlrpcval($this->dbname,      "string"));  
82         $msg->addParam(new xmlrpcval($this->id,          "int"));  
83         $msg->addParam(new xmlrpcval($this->password,    "string"));  
84         $msg->addParam(new xmlrpcval($relation,          "string"));  
85         $msg->addParam(new xmlrpcval("read",              "string"));  
86         $msg->addParam(new xmlrpcval($id_val,            "array"));  
87         $msg->addParam(new xmlrpcval($fields_val,        "array"));  
88  
89         $conn      = new xmlrpc_client($this->server_url . 'object');  
90         $conn->return_type = 'phpvals';  
91  
92         $resp = $conn->send($msg);  
93  
94         if ($resp->faultCode()){  
95             var_dump($resp);  
96             return -1;  
97         }  
98         else  
99             return ( $resp->value() );  
100    }  
101 }
```

Gambar 249 Method read()

Disini kita lakukan request read data dengan membentuk dulu object `$msg` yang merupakan class XML RPC message, `xmlrpcmsg('execute')`. Maksudnya adalah kita akan men-execute sebuah function pada suatu object.

Lalu kita tambahi parameter `$msg` dengan `$this->dbname`, `$this->user`, dan `$this->password`, yang terlebih dulu dibentuk menjadi object XML RPC value, `xmlrpcval`, type-nya string.

Parameter selanjutnya adalah nama relasi object yang akan di-search. Dalam hal ini kita jadikan input parameter `$relation`.

Parameter berikutnya adalah nama function yang akan di-execute yaitu "read".

Parameter berikutnya adalah id record yang akan dicari dalam bentuk XML-RPC array. Disini kita udah set pada variable \$id_val.

Parameter berikutnya adalah field-field dari record yang akan dicari dalam bentuk XML-RPC array. Disini kita udah set pada variable \$fields_val.

Variabel \$id_vals harus dibentuk terlebih dahulu agar berbentuk array XML-RPC berdasarkan array PHP pada input parameter \$ids.

Berikut ini cara membentuknya.

```
$id_val=array();
foreach ($ids as $i) {
    $id_val[] = new xmlrpcval($i , "int");
}
```

Disini kita hanya melakukan looping terhadap array PHP \$ids, dan pada setiap looping kita kumpulkan didalam array \$id_vals yang setiap elemennya adalah object xmlrpcval dari nilai integer setiap \$ids.

Variabel \$fields_vals juga harus dibentuk terlebih dahulu agar berbentuk array XML-RPC berdasarkan array PHP pada input parameter \$fields.

Berikut ini cara membentuknya.

```
$fields_val=array();
foreach ($fields as $f) {
    $fields_val[] = new xmlrpcval($f, "string");
}
```

Disini kita juga melakukan looping terhadap array PHP \$fields, dan pada setiap looping kita kumpulkan didalam array \$fields_vals yang setiap elemennya adalah object xmlrpcval bertipe "string" dari nilai setiap \$fields.

Test method read dengan menjalankan script lanjutan method search() seperti berikut :

```
193 $openerp = new MyOpenERPLib;
194 $userId = $openerp->login();
195 if ($userId == -1) {
196     die("Login error ....");
197 }
198
199 echo "=====\\Searching data containing java.....\n";
200 $session_ids = $openerp->search("academic.session","java");
201 var_dump($session_ids);
202
203 echo "=====\\Reading data containing java.....\n";
204 $fields      = array("name","duration");
205 $records     = $openerp->read("academic.session" , $session_ids, $fields);
206 foreach ($records as $key => $value) {
207     echo $value["name"];
208     echo "\n";
209 }
210
211
```

Gambar 250 Test method read()

Disini kita jalankan method read() dengan mengambil nilai array \$session_ids dari hasil pencarian sebelumnya melalui method search().

Return value dari method read() adalah array dari record yang berhasil dibaca sesuai id yang dimasukkan. Array ini memiliki key sesuai dengan nilai \$fields yang dikeluarkan pada saat read.

29.7 CREATE

Lanjut, bikin method create() pada class MyodooLib.
Deklarasinya seperti ini...

```
105     function create($relation, $values){
106
107         $msg = new xmlrpcmsg('execute');
108         $msg->addParam(new xmlrpcval($this->dbname, "string"));
109         $msg->addParam(new xmlrpcval($this->id, "int"));
110         $msg->addParam(new xmlrpcval($this->password, "string"));
111         $msg->addParam(new xmlrpcval($relation, "string"));
112         $msg->addParam(new xmlrpcval("create", "string"));
113         $msg->addParam(new xmlrpcval($values, "struct"));
114
115
116         $conn      = new xmlrpc_client($this->server_url . 'object');
117         $conn->return_type = 'phpvals';
118         $resp = $conn->send($msg);
119
120         if ($resp->faultCode()){
121             var_dump($resp);
122             return -1;
123         }
124         else{
125             return $resp->value();
126         }
127     }
128 }
```

Gambar 251 Method create()

Disini kita lakukan request create data dengan membentuk dulu object `$msg` yang merupakan class XML RPC message, `xmlrpcmsg('execute')`. Maksudnya adalah kita akan men-execute sebuah function pada suatu object.

Lalu kita tambahi parameter `$msg` dengan `$this->dbname`, `$this->id`, dan `$this->password`, yang terlebih dulu dibentuk menjadi object XML RPC value, `xmlrpcval`, type-nya string.

Parameter selanjutnya adalah nama relasi object yang akan di-create recordnya. Dalam hal ini kita jadikan input parameter `$relation`.

Parameter berikutnya adalah nama function yang akan di-execute yaitu "create".

Parameter berikutnya adalah array XML-RPC yang berisi nilai field-field yang akan di-insert yang diambil langsung dari input parameter variable `$values`. Variabel ini harus berupa array ketika method dijalankan.

Script untuk test method create.

```
194 $openerp = new MyOpenERPLib;
195 $userId = $openerp->login();
196 if ($userId == -1) {
197     die("Login error ....");
198 }
199
200 echo "=====\\nInserting data.....\\n";
201 $values = array(
202     'name' => new xmlrpcval("Session test dari XML","string"),
203     'start_date' => new xmlrpcval("2014-09-01","string")
204 );
205 $ret = $openerp->create("academic.session", $values);
206 echo $ret ;
207 echo "\\n";
208
```

Gambar 252 Test method create()

Disini kita bentuk dulu array input \$values yang berupa array dengan elemen object-object xmlrpcval. Key dari array adalah nama field dan values nya berupa xmlrpcval dari data yang akan diinsert, ber-type sesuai dengan type field masing-masing.

Array \$values kemudian dijadikan input parameter untuk method create(). Jika berhasil maka return value nya adalah id dari record yang berhasil di-create.

29.8 DELETE

Lanjut, bikin method delete() pada class MyodooLib.

Deklarasinya seperti ini...

```

130
131     function delete($relation, $ids) {
132         $id_val=array();
133         foreach ($ids as $i) {
134             $id_val[] = new xmlrpcval($i , "int");
135         }
136
137         $msg = new xmlrpcmsg('execute');
138         $msg->addParam(new xmlrpcval($this->dbname,      "string"));
139         $msg->addParam(new xmlrpcval($this->id,          "int"));
140         $msg->addParam(new xmlrpcval($this->password,    "string"));
141         $msg->addParam(new xmlrpcval($relation,          "string"));
142         $msg->addParam(new xmlrpcval("unlink",            "string"));
143         $msg->addParam(new xmlrpcval($id_val,             "array"));
144
145         $conn      = new xmlrpc_client($this->server_url . 'object');
146         $conn->return_type = 'phpvals';
147
148         $resp = $conn->send($msg);
149
150         if ($resp->faultCode()){
151             var_dump($resp);
152             return -1;
153         }
154         else
155             return ( $resp->value() );
156     }
157

```

Gambar 253 Method delete()

Disini kita lakukan request create data dengan membentuk dulu object `$msg` yang merupakan class XML RPC message, `xmlrpcmsg('execute')`. Maksudnya adalah kita akan men-execute sebuah function pada suatu object.

Lalu kita tambahi parameter `$msg` dengan `$this->dbname`, `$this->id`, dan `$this->password`, yang terlebih dulu dibentuk menjadi object XML RPC value, `xmlrpcval`, type-nya string.

Parameter selanjutnya adalah nama relasi object yang akan di-delete recordnya. Dalam hal ini kita jadikan input parameter `$relation`.

Parameter berikutnya adalah nama function yang akan di-execute yaitu "unlink".

Parameter berikutnya adalah id record yang akan dicari dalam bentuk XML-RPC array. Disini kita udah set pada variable `$id_val`.

Pembentukan `$id_val` sama seperti pada method `read()`.

Script untuk test method delete.

```
193 $openerp = new MyOpenERPLib;
194 $userId = $openerp->login();
195 if ($userId == -1) {
196     die("Login error ....");
197 }
198
199
200 echo "=====\\nDeleting data id=1.....\\n";
201 $ids = array(1);
202 $ret = $openerp->delete("academic.session", $ids );
203 echo $ret ;
204 echo "\\n";
205
```

Gambar 254 Test method delete()

Disini kita bentuk dulu array \$ids yang berisi id yang akan di-delete, contohnya 1. Array ini dijadikan input parameter bagi method delete().

Return value method ini berupa id yang berhasil di delete.

29.9 WRITE

Lanjut, bikin method write() pada class MyodooLib. Deklarasinya seperti ini...

```

159     function write($relation, $ids, $values ) {
160
161         global $conn, $dbname, $id, $password, $server_url;
162
163         $id_val=array();
164         foreach ($ids as $i) {
165             $id_val[] = new xmlrpcval($i , "int");
166         }
167
168         $msg = new xmlrpcmsg('execute');
169         $msg->addParam(new xmlrpcval($dbname, "string"));
170         $msg->addParam(new xmlrpcval($id, "int"));
171         $msg->addParam(new xmlrpcval($password, "string"));
172         $msg->addParam(new xmlrpcval($relation, "string"));
173         $msg->addParam(new xmlrpcval("write", "string"));
174         $msg->addParam(new xmlrpcval($id_val, "array"));
175         $msg->addParam(new xmlrpcval($values, "struct"));
176
177         $conn      = new xmlrpc_client($server_url . 'object');
178         $conn->return_type = 'phpvals';
179
180         $resp = $conn->send($msg);
181
182         if ($resp->faultCode()){
183             var_dump($resp);
184             return -1;
185         }
186         else
187             return ( $resp->value() );
188
189     }
190

```

Gambar 255 Method write()

Disini kita lakukan request write data dengan membentuk dulu object `$msg` yang merupakan class XML RPC message, `xmlrpcmsg('execute')`. Maksudnya adalah kita akan men-execute sebuah function pada suatu object.

Lalu kita tambahi parameter `$msg` dengan `$this->dbname`, `$this->id`, dan `$this->password`, yang terlebih dulu dibentuk menjadi object XML RPC value, `xmlrpcval`, type-nya string.

Parameter selanjutnya adalah nama relasi object yang akan di-delete recordnya. Dalam hal ini kita jadikan input parameter `$relation`.

Parameter berikutnya adalah nama function yang akan di-execute yaitu "write".

Parameter berikutnya adalah id record yang akan dicari dalam bentuk XML-RPC array. Disini kita udah set pada variable \$id_val.

Pembentukan \$id_val sama seperti pada method read().

Parameter berikutnya adalah array XML-RPC yang berisi nilai field-field yang akan di-insert yang diambil langsung dari input parameter variable \$values. Variabel ini harus berupa array ketika method dijalankan.

Script test method write():

```
193 $openerp = new MyOpenERPLib;
194 $userId = $openerp->login();
195 if ($userId == -1) {
196     die("Login error ....");
197 }
198
199 echo "=====\\nUpdating data id=2.....\\n";
200 $ids = array(2);
201 $values = array(
202     'name' => new xmlrpcval("Session update dari XML","string"),
203     'start_date' => new xmlrpcval("2014-09-01","string")
204 );
205
206 $ret = $openerp->write("academic.session", $ids, $values);
207 echo $ret ;
208 echo "\\n";
209
```

Gambar 256 Test method write()

Disini kita bentuk dulu array \$ids yang berisi id yang akan di-delete, contohnya 2. Array ini dijadikan input parameter bagi method write().

Lalu kita bentuk array input \$values yang berupa array dengan elemen object-object xmlrpcval. Key dari array adalah nama field dan values nya berupa xmlrpcval dari data yang akan di-update, ber-type sesuai dengan type field masing-masing.

Return value method adalah id record yang berhasil di-update.

29.10 WRITE ONE2MANY FIELDS

Bagaimana cara men-update atau meng-insert data berikut relasi one2many nya sekaligus ? jadi misanlya Session dan Attendee, gimana cara menginput Attendee sekaligus barengan dengan Session, tanpa harus insert Attendee satu-per-satu.

Berikut ini scriptnya.

```
200 echo "=====\\Create one2many data .....\\n";
201 $partners = array(
202     array("id"=>7, "name" => "nomor1"),
203     array("id"=>168, "name" => "nomor3"),
204     array("id"=>172, "name" => "joko"),
205 );
206 $att_lines = array();
207 foreach($partners as $p) {
208     $att_lines[] = new xmlrpcval(
209         array(
210             new xmlrpcval(0,'int'),
211             new xmlrpcval(0,'int'),
212             new xmlrpcval(
213                 array(
214                     'partner_id'=> new xmlrpcval($p["id"], 'string'),
215                     'name'      => new xmlrpcval($p["name"], 'string'),
216                 ),
217                 "struct"
218             )
219         ),
220         "array"
221     ); // one record
222 }
223
224 $attendee_ids = new xmlrpcval(
225     $att_lines,
226     "array"
227 );
228
229 $values = array(
230     'name'          => new xmlrpcval("Session Create o2m dari XML","string"),
231     'start_date'    => new xmlrpcval("2014-09-02","string"),
232     'attendee_ids'  => $attendee_ids
233 );
234
235
236 $ret = $openerp->create("academic.session", $values);
237 echo $ret ;
```

Gambar 257 Write field one2many

Pada intinya kita menggunakan method `write()` yang sama dengan sebelumnya.

Tapi disini kita update field `attendee_ids` yang merupakan field one2many pada Session object.

Untuk meng-insert atau update field jenis ini diperlukan nilai one2many commands, tapi dalam bentuk XML-RPC.

Kalau di Python kita cuman set data nya seperti ini:

```
[ (0,0,{data1}) , (0,0,{data2}) , ... ]
```

Di PHP XML-RPC data itu harus dibentuk dengan logika seperti ini:

Misalnya kita punya data partner array seperti ini ingin dimasukkan sebagai Attendee suatu Session.

```
$partners = array(
    array("id"=>7, "name" => "nomor1"),
    array("id"=>168, "name" => "nomor3"),
    array("id"=>172, "name" => "joko"),
);
```

Array partner itu harus di-looping dalam rangka membentuk array \$attendee_ids dalam format one2many command.

Pertama kita kumpulin di variable array `$att_lines` object `xmlrpcval` berjenis array yang isinya adalah array dari 0,0, dan data. Data adalah object `xmlrpcval` berjenis "struct" yang terdiri dari array dengan key '`'partner_id'`' dan '`'name'`' dan value-nya adalah id partner dan nama partner dalam bentuk `xmlrpcval` berjenis string.

```
$att_lines = array();
foreach($partners as $p) {
    $att_lines[] = new xmlrpcval(
        array(
            new xmlrpcval(0,'int'),
            new xmlrpcval(0,'int'),
            new xmlrpcval(
                array(
                    'partner_id' => new
xmlrpcval($p["id"], 'string'),
                    'name'          => new
xmlrpcval($p["name"], 'string'),
                ),
                "struct"
            )
        ),
        "array"
    ); // one record
```

```
}
```

Setelah terkumpul sesuai dengan banyaknya array partner, variable `$att_lines` dibentuk lagi menjadi variable `$attendee_ids` yang merupakan object `xmlrpcval` ber-type array.

```
$attendee_ids = new xmlrpcval(
    $att_lines,
    "array"
);
```

Baru kemudian variable `$attendee_ids` dimasukkan sebagai nilai dari field `attendee_ids` object session. Semua field dijadikan array `$values` yang siap dikirim ke method `create()` atau `update()`.

```
$values = array(
    'name'          => new xmlrpcval("Session Create o2m dari
XML", "string"),
    'start_date'   => new xmlrpcval("2014-09-02", "string"),
    'attendee_ids' => $attendee_ids
);
$ret = $odoo->create("academic.session", $values);
```

30 WEB SERVICE DENGAN PHP RIPCORD

30.1 PERSIAPAN

Buat folder di **htdocs/odoo-client**.

Download Ricord library , simpan di **odoo-client/ripcord**.

Dalamnya ada file **ripcord.php**.

Ricord Library URL: <https://github.com/poef/ripcord>

Dokumentasi API web service Odoo:

https://www.odoo.com/documentation/10.0/api_integration.html

30.2 INCLUDE LIBRARY RIPCORD DI SCRIPT PHP

Buat file **test.php** dan include library ricord. Caranya:

```
<?php  
include ("ripcord/ripcord.php");  
  
$url    = "http://localhost:8069";  
$username = "admin";  
$password = "1";  
$dbname = "academic";
```

30.3 PROSES LOGIN

Login dari PHP ke Odoo dengan user dan password Odoo user.

Sebelumnya kita harus membentuk object \$common untuk object client XMLRPC pada alamat /xmlrpc/2/common untuk proses login ke Odoo.

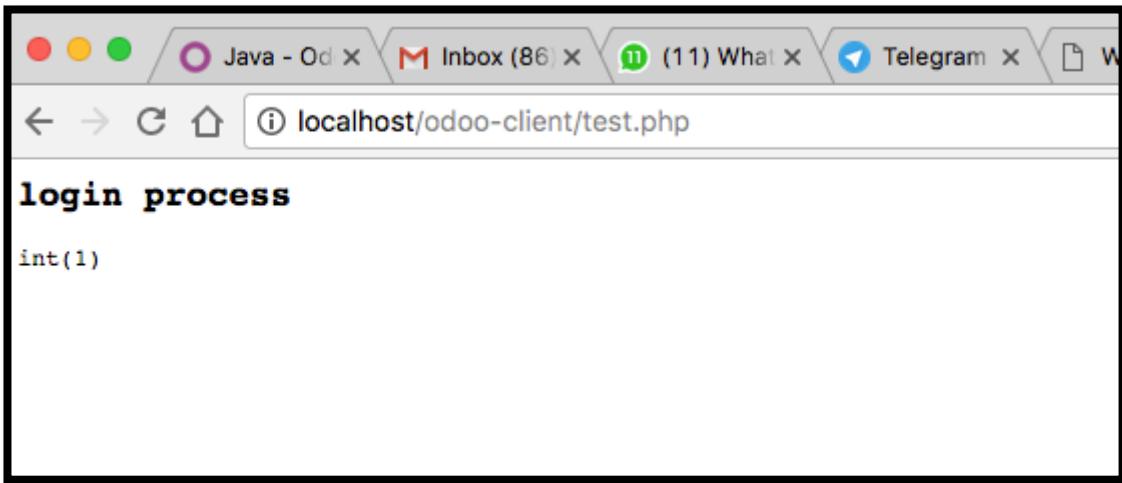
Lalu jalankan method authenticate untuk login user.

Caranya:

```
echo "<h2>login process</h2>";  
  
$common = ripcord::client("$url/xmlrpc/2/common");  
$uid = $common->authenticate($dbname, $username, $password, array());  
var_dump($uid);
```

Testing dari browser: <http://localhost/odoo-client/test.php>

Variabel \$uid: integer ID user jika berhasil login atau False jika gagal login.



Gambar 258 Proses login

30.4 PROSES SEARCH DAN READ DATA

Mencari data Course berdasarkan nama. Cari dulu ID record dengan method **search** lalu jalankan method **read**.

Sebelumnya kita harus membentuk object \$model untuk object client XMLRPC pada alamat **/xmlrpc/2/object**.

```
/* create $models object*/
$models = ripcord::client("url/xmlrpc/2/object");

echo "<h2>search and then read academic.course</h2>";

/* search ID partner */
$ids = $models->execute_kw($dbname, $uid, $password,
    'academic.course', 'search',
    array(
        array(
            array('name', '=', 'Java')
        )
    )
);
var_dump($ids);

/* read records by ID */
$records = $models->execute_kw($dbname, $uid, $password,
    'academic.course', 'read', array( $ids ));
```

```

echo "<br/>";

foreach ($records as $key => $value) {
    echo $value['name'] . "<br/>";
    echo $value['description'] . "<br/>";
}

```

Hasilnya:

```

login process

int(1)

search and then read academic.course

array(1) {
  [0]=>
  int(3)
}

Java
Kursus Java

```

Atau bisa menggunakan method **search_read** langsung:

```

/* search and read */

echo "<h2>search_read academic.course</h2>";

$records = $models->execute_kw($dbname, $uid, $password,
    'academic.course', 'search_read',
    array(
        array(
            array('name', 'ilike', 'java'),
        )
    )
);

echo "<br/>";

foreach ($records as $key => $value) {
    echo $value['name'] . "<br/>";
    echo $value['responsible_id'][1] . "<br/>";
}

```

Hasilnya

```
login process

int(1)

search and then read academic.course

array(1) {
    [0]=>
    int(3)
}

Java
Kursus Java

search_read academic.course

Java
Andi
```

Disini kita menampilkan field many2one **responsible_id** dengan cara :

```
$value['responsible_id'][1]
```

karena field many2one berupa array yang terdiri dari 2 elemen yaitu ID dan name record terkait.

30.5 PROSES CREATE

Create data session pada course yang dipilih diatas. Buat file PHP baru: **create.php**.

Isinya

```
<pre>
<?php
include "ripcord/ripcord.php";
```

```

$url      = "http://localhost:8069";
$username = "admin";
$password = "1";
$dbname  = "academic";

/* login */
echo "<h2>login process</h2>";

$common = ripcord::client("$url/xmlrpc/2/common");
$uid = $common->authenticate($dbname, $username, $password, array());
var_dump($uid);

/* create $models object*/
$models = ripcord::client("$url/xmlrpc/2/object");

echo "<h2>create course</h2>";

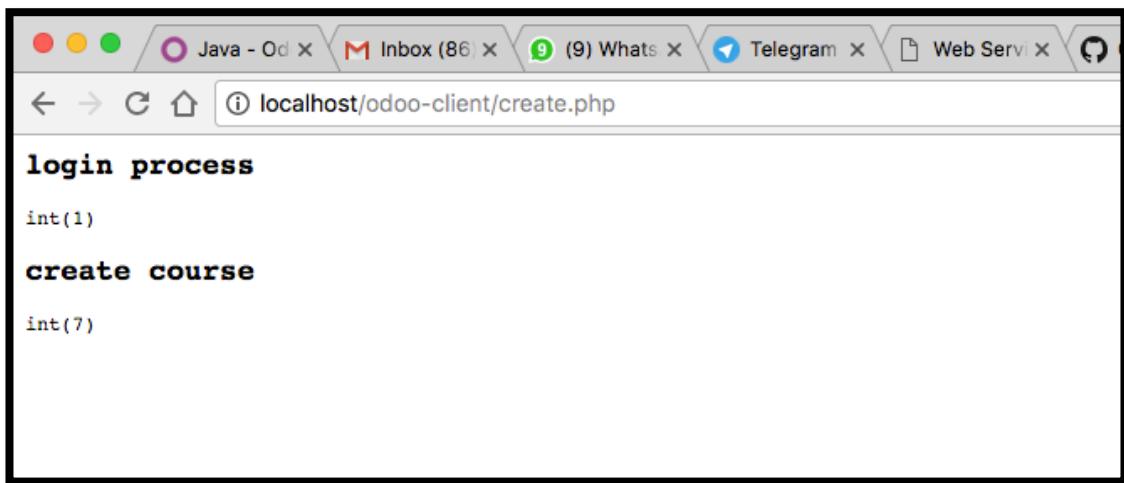
$id = $models->execute_kw($dbname, $uid, $password,
    'academic.course', 'create',
    array(
        array(
            'name'=>"PHP XMLRPC",
            'description'=>"Integrasi PHP dengan Odoo",
        ),
    )
);
var_dump($id);

```

Pada proses ini kita jalankan method create pada object yang akan dicreate recordnya, misalnya academic.course.

Lalu pada parameter method tersebut, kita sertakan nama field dan value-nya menggunakan array associative.

Hasilnya:



Gambar 259 Create course

pada Odoo akan terbentuk record Course yang dibuat melalui PHP.

Name	Description
Java	Kursus Java
Odoo	
Python	
Dotnet	
PHP XMLRPC	Integrasi PHP dengan Odoo

Gambar 260 Hasil create di Odoo

30.6 PROSES WRITE

Update data session yang sudah dcreate sebelumnya. Buat file PHP baru dengan nama: **write.php**.

Isinya:

```
<pre>
<?php
include "ripcord/ripcord.php";

$url = "http://localhost:8069";
$username = "admin";
$password = "1";
$dbname = "academic";

/* login */
echo "<h2>login process</h2>";

$common = ripcord::client("$url/xmlrpc/2/common");
$uid = $common->authenticate($dbname, $username, $password, array());
var_dump($uid);

/* create $models object*/
$models = ripcord::client("$url/xmlrpc/2/object");

echo "<h2>search and then write academic.course</h2>";

/* search ID course */
$id = $models->execute_kw($dbname, $uid, $password,
    'academic.course', 'search',
    array(
        array(
            'name' => 'Integrasi PHP dengan Odoo'
        )
    )
);</pre>
```

```

        array('name','=', 'Java'),
    )
);
var_dump($id);

echo "<h2>write course with id=$id[0] </h2>";

$ret = $models->execute_kw($dbname, $uid, $password,
    'academic.course', 'write',
    array(
        $id,
        array('description'=>"kursus java di update via PHP")
    )
);
var_dump($ret);

```

Hasilnya:

```

login process
int(1)

search and then write academic.course

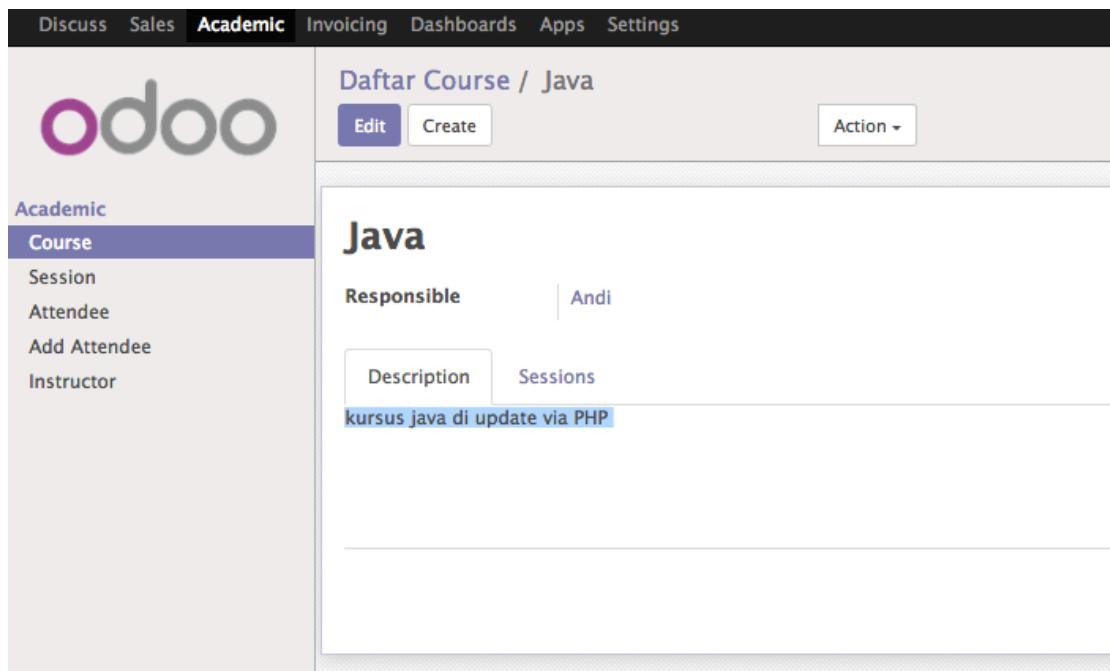
array(1) {
[0]=>
int(3)
}

write course with id=3

bool(true)

```

di Odoo, record tersebut akan terupdate:



30.7 PROSES DELETE

Delete data session berdasarkan nama. Buat file PHP baru dengan nama: **delete.php**.

Isinya:

```
<pre>
<?php
include "ripcord/ripcord.php";

$url    = "http://localhost:8069";
$username  = "admin";
$password = "1";
$dbname = "academic";

/* login */
echo "<h2>login process</h2>";

$common = ripcord::client("$url/xmlrpc/2/common");
$uid = $common->authenticate($dbname, $username, $password, array());
var_dump($uid);

/* create $models object*/
$models = ripcord::client("$url/xmlrpc/2/object");

echo "<h2>search and then delete academic.course</h2>";

/* search ID course */
$ids = $models->execute_kw($dbname, $uid, $password,
    'academic.course', 'search',
    array(
        array(
```

```

        array('name','=','PHP XMLRPC'),
    )
);
var_dump($ids);

echo "<h2>delete course with id=$ids[0] </h2>";

$ret = $models->execute_kw($dbname, $uid, $password,
    'academic.course', 'unlink',
    array(
        $ids,
    )
);
var_dump($ret);

```

Jalankan dan lihat hasilnya:

```

login process
int(1)
search and then delete academic.course
array(1) {
    [0]=>
    int(7)
}
delete course with id=7
bool(true)

```

Record tersebut akan hilang di Odoo.



30.8 PROSES CREATE ATAU WRITE FIELD ONE2MANY

Contoh mengupdate data attendee suatu session sekaligus dengan 1 command write ke session.

Buat file baru dengan nama update-attendee.php.

Isinya.

```
<pre>
<?php
include "ripcord/ripcord.php";

$url    = "http://localhost:8069";
$username  = "admin";
$password = "1";
$dbname = "academic";

/* login */
echo "<h2>login process</h2>";

$common = ripcord::client("$url/xmlrpc/2/common");
$uid = $common->authenticate($dbname, $username, $password, array());
var_dump($uid);

/* create $models object*/
$models = ripcord::client("$url/xmlrpc/2/object");

/* search ID partner Agus */
$partner1_id = $models->execute_kw($dbname, $uid, $password,
    'res.partner', 'search',
    array(
        array(
            array('name', '=', 'Agus'),
        )
    )
)
```

```

);

/* search ID partner Budi */
$partner2_id = $models->execute_kw($dbname, $uid, $password,
    'res.partner', 'search',
    array(
        array(
            array('name', '=', 'Budi'),
        )
    )
);

/* search ID session */
$ids = $models->execute_kw($dbname, $uid, $password,
    'academic.session', 'search',
    array(
        array(
            array('name', '=', 'Session 1 Java'),
        )
    )
);
var_dump($ids);

echo "<h2>write attendee_ids on academic.session</h2>";

$ret = $models->execute_kw($dbname, $uid, $password,
    'academic.session', 'write',
    array(
        $ids,
        array(
            'attendee_ids' =>array(
                array(0,0,array(
                    'partner_id'=> $partner1_id[0],
                    'name'=>"01")),
                array(0,0,array(
                    'partner_id'=> $partner2_id[0],
                    'name'=>"02"))
            )
        )
    )
);
var_dump($ret);

```

Untuk proses create, field attendee_ids persis sama formatnya.

Jalankan dan lihat hasilnya.

```

Session 1 × M Inbox (87) × (10) What × Telegram × Web Serv ×
localhost/odoo-client/write-attendee.php

login process

int(1)
array(1) {
    [0]=>
    int(4)
}

write attendee_ids on academic.session

bool(true)

```

Di Odoo session tersebut akan terisi attendee-nya.

Daftar Session / Session 1 Java

Course	Java	Duration	0						
Instructor		Seats	0						
Start Date	02/24/2017	Active	<input checked="" type="checkbox"/>						
Image Small		Taken Seat	0%						
Attendees <table border="1"> <thead> <tr> <th>Name</th> <th>Partner</th> </tr> </thead> <tbody> <tr> <td>01</td> <td>Agus</td> </tr> <tr> <td>02</td> <td>Budi</td> </tr> </tbody> </table>				Name	Partner	01	Agus	02	Budi
Name	Partner								
01	Agus								
02	Budi								

30.9 PROSES CREATE ATAU WRITE FIELD MANY2MANY

Menambah tags di instruktuktur. Buat file baru dengan nama **add-partner-tags.php**.

Isinya.

```

<pre>
<?php
include "ripcord/ripcord.php";

```

```

$url    = "http://localhost:8010";
$username = "admin";
$password = "1";
$dbname = "academic";

/* login */
echo "<h2>login process</h2>";

$common = ripcord::client("$url/xmlrpc/2/common");
$uid = $common->authenticate($dbname, $username, $password, array());
var_dump($uid);

/* create $models object*/
$models = ripcord::client("$url/xmlrpc/2/object");

/* create tag 1 */
$tag1_id = $models->execute_kw($dbname, $uid, $password,
    'res.partner.category', 'create',
    array(
        array(
            'name'      => "Senior",
        ),
    )
);
var_dump($tag1_id);

/* create tag 2 */
$tag2_id = $models->execute_kw($dbname, $uid, $password,
    'res.partner.category', 'create',
    array(
        array(
            'name' => 'Part Time',
        )
    )
);
var_dump($tag2_id);

/* search ID instruktur */
$ids = $models->execute_kw($dbname, $uid, $password,
    'res.partner', 'search',
    array(
        array(
            array('name', '=', 'Joko'),
            array('is_instructor', '=', True)
        )
    )
);
var_dump($ids);
if (!$ids){
    die("instructor not found!");
}

echo "<h2>write category_id on res.partner</h2>";

$ret = $models->execute_kw($dbname, $uid, $password,
    'res.partner', 'write',
    array(
        $ids,
        array(
            'category_id'    =>array(
                array(4,$tag1_id),
                array(4,$tag2_id)
            )
        )
    )
);
var_dump($ret);

```

```
        )
    )
);
var_dump($ret);
```

Jalankan dan lihat hasilnya.

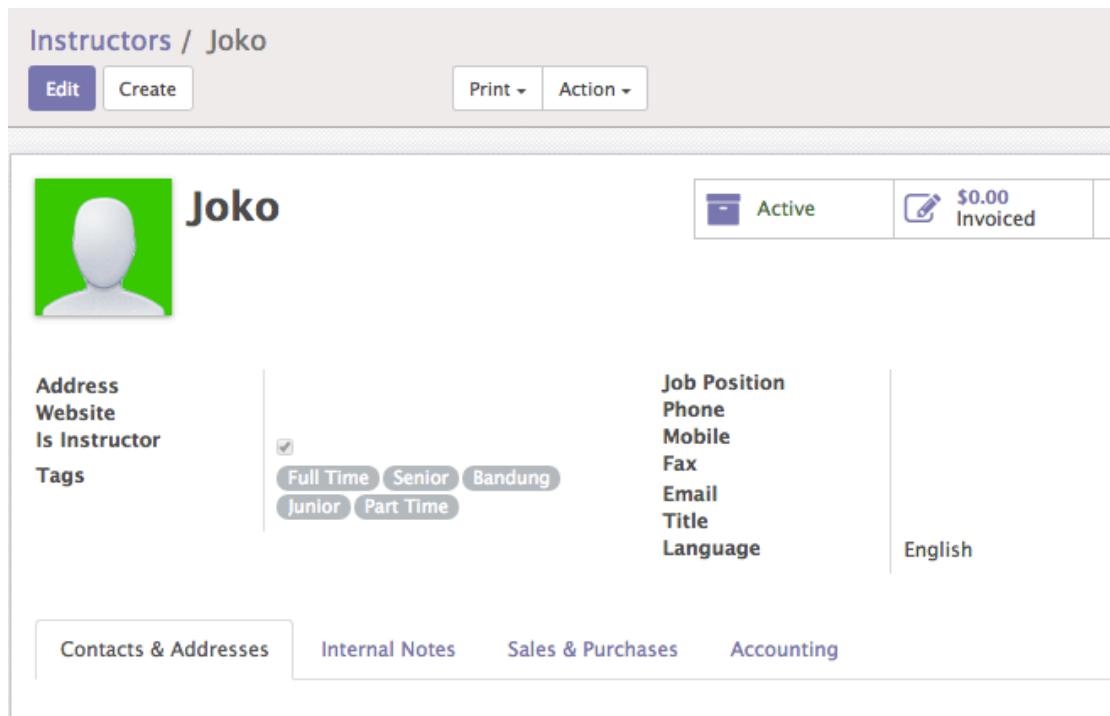
Instructors / Joko

Edit Create Print Action

 **Joko** Active \$0.00 Invoiced

Address Website Is Instructor Tags Job Position
Full Time Senior Bandung
Junior Part Time Phone Mobile Fax Email Title Language English

Contacts & Addresses Internal Notes Sales & Purchases Accounting



31 REKAP HARI 5

Report RML, install plugin OpenOffice, membuat template report baru, modif report lama, menyertakan di addons.

Report Webkit, instalasi, bikin report lewat user interface, bikin report lewat addons.

Dashboard, menampilkan graph, calendar, dan list.

Web service PHP XML-RPC, login, create, write, dan create dengan one2many fields.

32 PENUTUPAN

Download module yang udah jadi dari sini

https://www.dropbox.com/sh/2sdljmfzapmlacb/AAD3_J7aEDV-6FM7jjAFYCGGa?dl=0

33 REFERENSI

odoo Techincal Training v10

odoo Official Documentation [docs.odoo.com]

Python Official Documentation [docs.python.org]

34 TENTANG PENULIS

Akhmad Daniel Sembiring

2003 -- Now: CEO VITRAINING.COM

TEMPAT DAN TANGGAL LAHIR

Binjai, 19 September 1971

akhmad.daniel@gmail.com

PENDIDIKAN:

- | | |
|-------------|--|
| 1995 – 1998 | Program Magister Teknik Elektro Institut Teknologi Bandung |
| 1990 – 1994 | Teknik Elektro Institut Teknologi Bandung , Indonesia Sub Jurusan Teknik Sistem Komputer |

PENGALAMAN:

- 2014 Implementasi odoo Mutif**, odoo for garment business, purchase, manufacturing, sales, accouting.
- 2014 Implementasi odoo Koperasi Karyawan Freeport Indonesia**, odoo for retail business, point of sales antar cabang, movement antar gudang, purchase, sales, accounting.
- 2013 Mobile Sales, Logistic, and Support**, for Surya Manggala Informatika, aplikasi mobile Android dan Grails server untuk mobile sales, logistic, and engineer support
- 2013 Implementasi odoo Tiket.com**, integrasi portal e-commerce tiket.com dengan odoo untuk menghasilkan invoice, payment, journal sesuai dengan tipe barang, pembayaran, discount, dan partner.
- 2013 Pasific Prestress Indonesia PT**, implementasi odoo untuk modul HRD, data karyawan, recruitment, payroll, perhitungan lembur sesuai dengan peraturan Menteri Tenaga Kerja, import data absensi fingerprint.
- 2012 Asiades, Furniture Distributor Company, Dubai, UAE**. odoo implementation and customization for furniture distributor company, implementation of down payment journal where the Company can track Down Payment prior to invoice creation, and when the invoice is paid, the Down Payment is taken into account so the payment will be done only by the outstanding amount.
- 2012 SAGE Petroleum, Ghana, Africa**: odoo implementation and customization for petroleum distribution company for Product Discount Journal, Nominal Discount Entry, Discount per Customer per Product database, Analytical Account Setup, and Government Subsidized Product Sales Journal, project management, bill of material product costing based on raw material total cost
- 2012 Sunberry LZE and Breezer Petroleum Card System**: odoo implementation and customization to integrate to Petro Plus Card Management system for maintaining the loyalty card system, postpaid and prepaid card for petroleum payment system, project management
- 2011 Pengembangan aplikasi eBengkel untuk PT Pendopo Multikarya**, aplikasi akuntansi untuk bisnis bengkel online.
- 2011 Pengembangan aplikasi XLS Report Generator** for Deutsche Bank, bekerjasama dengan PT Skyworx Indonesia
- 2011 Pengembangan aplikasi Mobile Survey System** berbasis Android untuk aplikasi Loan Originating System, Bank Muamalat, bekerjasama dengan PT Skyowrx Indonesia
- 2011 SAGE Operational Management System**

- Membangun sistem management inventory bahan bakar minyak untuk SAGE Petroleum Limited, Ghana, Africa.
- 2011 Fixed Asset Management System Rabobank International Indonesia**
Membangun system management fixed asset, procurement, dan inventory untuk PT Rabobank International Indonesia.
- 2011 MyPushme.com**
Membangun sistem sinkronisasi data contact, email, picture antara mobile devices berbasis Symbian, Android, iPhone, dan Windows Mobile dengan MS Outlook, Gmail, Yahoo, POP3/IMAP, untuk Sunberry Limited, Ghana, Africa.
- 2010 Kukuloba.com**
Membangun sistem portal diskon. Website ini merupakan media dimana pelaku usaha dapat mengiklankan produk dan jasa secara gratis dengan cara memberikan diskon yang menarik dan menentukan jumlah minimal penjualan yang Anda inginkan. Disini kami menggunakan sistem penawaran harga diskon dalam batas waktu 24 jam. Artinya, harga diskon hanya akan berlaku jika jumlah minimal penjualan terpenuhi dalam jangka waktu tersebut. Jika belum terpenuhi maka penawaran diskon itu otomatis tidak berlaku.
- 2010 GPS Tracking Indonesia**
Membangun sistem pelacak kendaraan berbasis GPS dan GSM untuk memantau posisi kendaraan secara realtime melalui web dan mobile device.
- 2009 Ligarwangi.com**
Membangun sistem toko online ligarwangi.com yang dapat menerima pembayaran melalui transfer bank BCA, Mandiri, dan BNI, serta pembayaran melalui Credit Card melalui PayPal. Sistem ini dibangun menggunakan Zen Cart.
- 2009 IALF Jakarta**
Membangun sistem SMS Gateway yang dapat menangani pengiriman SMS melalui Microsoft Outlook 2007 dengan menggunakan Outlook Mobile Services (OMS). SMS dikirimkan melalui HTTP dan serial port/ modem GSM
- 2009 Direktorat Jenderal MIGAS Jakarta**
Membangun sistem integrasi antara sistem monitoring realisasi anggaran (DIPA) dengan sistem Aplikasi Surat Perintah Membayar (SPM 2000) Departemen Keuangan. Aplikasi ini menjembatani proses pembuatan SPM secara online antara sistem aplikasi monitoring anggaran yang dibuat diatas MS SQL Server dengan Aplikasi SPM2009 yang dibangun di atas database FoxPro.
- 2009 IALF Jakarta**
Membangun sistem aplikasi Fixed Asset Management System, Purchasing, dan Inventory System, yang dimulai dari proses requisition, purchase order, delivery order, invoice, bank disbursement voucher untuk mencetak ke cek dan giro bank, serta sistem inventory management.
- 2009 Heliantono dan Rekan, Kantor Akuntan Publik**
Audit sistem informasi berbasis Oracle pada Institut Teknologi Bandung (ITB) sebagai bagian dari audit sistem keuangan yang dilakukan oleh Kantor Akuntan Publik untuk tahun laporan 2008.
- 2009 IALF Jakarta**
Membangun sistem aplikasi pembuatan user Active Directory yang mampu men-create user pada server Active Directory Windows 2003 dengan kriteria tertentu dan jumlah user tertentu, mencetak label user dan pasword untuk digunakan sebagai access code untuk mengakses internet di jaringan hot spot IALF Jakarta.
- 2008 PT Bank Bumiputera, Tbk**

- Membangun sistem aplikasi Fixed Asset Management System, yang dimulai dari proses permohonan dan persetujuan pembelian, purchase order, delivery order, invoice, dan integrasi dengan Silverlake melalui GL voucher.
- 2008 IALF Surabaya**
Membangun sistem aplikasi multimedia menggunakan Google Desktop. Pada aplikasi ini user dapat mencari bahan-bahan pelajaran baik berupa video, gambar, text, document, dan lain sebagainya melalui sebuah interface yang mudah digunakan dengan back end indexing Google Desktop. Materi tersebut disimpan pada sebuah server yang dapat diindex oleh Google Desktop. Ketika sebuah file materi ditemukan, user dapat mengakses file tersebut melalui Windows sharing sehingga misalnya materi dalam bentuk DVD (yang telah dibuat menjadi ISO file) dapat dimainkan pada komputer local seolah-olah sedang menjalankan DVD pada sebuah DVD Drive yang memungkinkan menu dan semua fitur DVD lainnya aktif. Pada komputer local terdapat program mounter yang melakukan mounting file dengan extension ISO menjadi sebuah virtual drive.
- 2008 Kementerian Negara BUMN**
Bekerjasama dengan PT INTI Bandung, membangun sistem aplikasi Executive Information System (EIS) yaitu sistem pelaporan data keuangan dan operasional 139 BUMN ke Kementerian Negara BUMN secara online.
- 2007 PT Indonesia Media Technology**
Pengembangan Sistem Aplikasi SMS Gateway untuk portal Astaga.com
- 2007 PT Indonesia Media Technology**
Pengembangan Sistem Portal Astaga.com versi 2, pengembangan dan redesign portal lama, penambahan fitur dan channel baru.
- 2006 PT Telekomunikasi Indonesia**
Bekerjasama dengan PT INTI Bandung, membangun sistem aplikasi Konversi Call Data Record (CDR) menjadi file ASCII untuk sentral Siemens EWSD, AT&T, Samsung, dan NEAX
- 2006 Aplikasi Sistem Perkantoran Elektronis PT ASDP Indonesia Ferry**
Implementasi aplikasi Sistem Perkantoran Elektronis, meliputi pembuatan surat-surat dinas, penomoran, klasifikasi surat, arsip, penerimaan surat, scanning, email, printing, dll.
- 2006 Aplikasi SMS Gateway TOLL SMS 5272**
Implementasi aplikasi SMS Gateway untuk informasi kemacetan jalan tol, quiz pengguna jalan tol, zodiak, berita, dll. Bekerjasama dengan PT MIK, DIME, dan Jasa Marga.
- 2006 Project Management Office, PT TELKOM Divisi ISC**
Implementasi aplikasi Project Management Office, bekerja sama dengan PT INTI. Menggunakan software open source dotproject, meliputi instalasi, migrasi data, koneksi dengan LDAP Server PT TELKOM, dan training user dan administrator.
- 2005 Billing ISP -- Jalawave Cakrawala Bandung**
Implementasi aplikasi Billing ISP untuk perusahaan internet service provider. Terdiri dari billing untuk customer postpaid maupun prepaid. Realtime checking ke network access server. Menggunakan freeRADIUS, freeside, MySQL , Postgres, Perl, dan Apache sebagai infrastruktur aplikasinya.
- 2005 Aplikasi SMS Gateway, PT KAESKA Media**
Implementasi aplikasi SMS Gateway untuk PT KAESKA Media - sebuah perusahaan content provider - untuk aplikasi SMS interaktif informasi kampus, sekolah, polling, info harga dan kurs, dan sebagainya.

Termasuk di dalamnya adalah aplikasi download ringtone, wallpaper, logo, java games.

2005 Outsourcing untuk Ghana Health Service - Ghana , Africa

Modifikasi dan integrasi beberapa program opensource untuk aplikasi sistem informasi rumah sakit dan financial terintegrasi. Di sini client menginginkan agar program open source Care2x dan SQL Ledger dapat saling terintegrasi dan dapat sesuai dengan logika bisnis yang di Ghana Africa.

2005 Sistem Informasi Akademik/ Politeknik Negeri Medan

Software untuk automatisasi sistem administrasi akademik Politeknik Negeri Medan. Mencakup sistem Registrasi, Kenaikan Kelas, Absensi, Biodata mahasiswa dan dosen, daftar nilai, KRS, Transkrip Nilai (Mark Sheet), KTM, Pembayaran SPP, Yudisium, Surat Peringatan, dan lain-lain.

2005 Fixed Asset Management System I/A/L/F Denpasar

Software untuk otomatisasi proses administrasi dan manajemen Fixed Asset (Aktiva Tetap) di Indonesia Australia Language Foundation Jakarta. Sistem ini meliputi proses pembelian, disposal, penjualan, penghapusan, pemindahan, physical inventory, printing barcode, dll.

Laporan yang dihasilkan adalah laporan aktiva tetap, laporan penyusutan, laporan pembelian, penjualan, disposal, dan sebagainya.

2005 Academic Finance Administration System I/A/L/F Surabaya

Software untuk otomatisasi proses administrasi dan keuangan di Indonesia Australia Language Foundation Surabaya.

Sistem ini meliputi proses registrasi peserta kursus dan IELTS Test, pembayaran (cash, credit card, bank transfer, cheque) , manajemen peserta kursus dan test (pindah jadwal, refund, cancel), manajemen kelas, client, dan teacher, sistem administrasi manajerial, sistem administrasi keuangan, manajemen inventory buku dan merchandising.

2005 POS PBB BALIKPAPAN

Bekerjasama dengan salah satu perusahaan IT di Balikpapan untuk desain dan implementasi sistem online pembayaran Pajak Bumi dan Bangunan (PBB) kota Balikpapan.

Mencakup 12 bank Tempat Pembayaran (TP), 1 kantor Pelayanan Pajak Bumi dan Bangunan (KP PBB), 1 kantor Dinas Pendapatan Daerah.

2005 Network Monitoring System, PT TELKOM

Bekerjasama dengan PT CMI dan ITB untuk mendesain dan implementasi sistem Network Management System, menggantikan sistem lama yang menggunakan Alcatel. Memonitor jaringan microwave Jakarta-Bandung (Dayeuh Kolot), menampilkan tampilan dalam bentuk web interface, sehingga dapat dimonitor di kantor pusat PT TELKOM.

2004 Academic Finance Administration System I/A/L/F Denpasar

Software untuk otomatisasi proses administrasi dan keuangan di Indonesia Australia Language Foundation Denpasar Bali.

Sistem ini meliputi proses registrasi peserta kursus dan IELTS Test, pembayaran (cash, credit card, bank transfer, cheque) , manajemen peserta kursus dan test (pindah jadwal, refund, cancel), manajemen kelas, client, dan teacher, sistem administrasi manajerial, sistem administrasi keuangan, manajemen inventory buku dan merchandising.

Mencakup juga sistem Customer Relationship Management (CRM) untuk mencatat progres follow up proposal ke client yang meminta training inhouse, Inventory Fixed Asset, dan Human Resource Database.

2004 Academic Finance Administration System I/A/L/F Surabaya

Pengembangan Sistem Informasi Administrasi Kursus untuk Indonesia Australia

- Language Foundation / IALF – Surabaya.
 Sistem ini meliputi proses registrasi peserta kursus dan IELTS Test, pembayaran (cash, credit card, bank transfer, cheque), manajemen peserta kursus dan test (pindah jadwal, refund, cancel), manajemen kelas, client, dan teacher, sistem administrasi manajerial, sistem administrasi keuangan, manajemen inventory buku dan merchandising.
- 2003 Pengembangan Sistem Informasi Administrasi Kursus untuk **Indonesia Australia Language Foundation / IALF** – Jakarta
 Sistem ini meliputi proses registrasi peserta kursus dan IELTS Test, pembayaran (cash, credit card, bank transfer, cheque), manajemen peserta kursus dan test (pindah jadwal, refund, cancel), manajemen kelas, client, dan teacher, sistem administrasi manajerial, sistem administrasi keuangan, manajemen inventory buku dan merchandising.
- 2003 Pengembangan sistem informasi **Community Development PT Riau Andalan Pulp and Paper**, Pelalawan, Riau, PT MYOH Technology
- 2003 Pengembangan sistem informasi **Jaminan Pelayanan Kesehatan Masyarakat (JPKM) di Kimia Farma Persero**, Jakarta, PT MYOH Technology
- 2003 Pengembangan sistem pembayaran Pajak Bumi dan Bangunan (PBB) Online di Kotamadya Tangerang, Banten, PT MYOH Technology
- 2001 Pembuatan layanan SMS Gateway – PT Celicom Indonesia
- 2000 **System Architect**, PT Celicom Indonesia
- 1998 **PoInter Internet Training Center**, koordinator dan staf pengajar pelatihan di bidang teknologi informasi.
- 1998 **PT Semen Gresik**, Konsultan Pengembangan Sistem Informasi
- 1998 **PT Semen Gresik**, Konsultan Penanganan Y2K untuk Sistem Informasi
- 1997 **PoInter**, Pojok Internet, warung internet di Bandung.
- 1996 **Badan Pengelola Dampak Lingkungan (BAPEDAL) Jakarta** Perancangan dan Instalasi Internet Server untuk Sistim informasi Lingkungan Ekonet-Bapedal dan interkoneksi jaringan lokal ke Internet
- 1996 **Direktorat Teknologi PT.Krakatau Steel Cilegon** Perancangan dan Instalasi Secure LAN (Firewall), Aplikasi Internet dan interkoneksi ke jaringan Internet melalui VSAT
- 1996 **Lembaga Pendidikan Perkebunan (LPP) Yogyakarta** Supervisi untuk pengembangan beberapa layanan Internet pada LAN dan Novell Netware untuk LAN dan WAN
- 1996 **Indonesia OnLine Access (IdOLA) , PT. Aplikanusa Lintasarta Jakarta** Konsultasi pengembangan layanan Internet dan PPP billing system untuk Network operation Center (NOC) IdOLA sebagai Internet Service Provider
- 1994 **Polytechnique Education Development Center (PEDC)** Project Engineer pada pengembangan sistem jaringan komputer paket radio TCP/IP menggunakan HF link untuk 6 Politeknik di Indonesia.
- 1994 **PT. Elektrindo Nusantara & Computer Network Research Group** Staf peneliti pada proyek pengembangan HDLC card untuk sistem komunikasi data berkecepatan tinggi berbasis TCP/Ip via Satelit
- 1994 **Jurusan Teknik Elektro Institut Teknologi Bandung** Pengembangan Jaringan Backbone Jurusan teknik Elektro dan ITB-NET
- 1993 **Pusat Antar Universitas bidang Mikroelektronika (PAUME)** Institut Teknologi Bandung, Staf Peneliti pada Computer Network Rearch Group

PENGALAMAN MENGAJAR :

- 2008 **Kementerian Negara BUMN**

- Pelatihan aplikasi Portal Executive Information System untuk 139 BUMN seluruh Indonesia
- 2006 **PT ASDP Indonesia Ferry**
Training HTML , PHP, MySQL tahap II untuk calon staf IT PT ASDP seluruh Indonesia.
- 2006 **PT ASDP Indonesia Ferry**
Training HTML , PHP, MySQL tahap I untuk calon staf IT PT ASDP seluruh Indonesia.
- 2004 **PT Telkom VI Balikpapan** – Pendidikan Profesi Bidang Teknologi Informasi, Teknologi Wide Area Network, LAPI ITB
- 1999 **Pertamina UP III Plaju**, training Active Server Pages
- 1998 **PoInter Internet Training Center**, staf pengajar
- 1998 **In-House Training Pengembangan Sistem Intranet** untuk staf PT. Semen Gresik, Tretes
- 1997 **In-House Training Computer Networking & Data Communication** untuk staf PT. Semen Padang, Semen Gresik & Semen Tonasa, Padang, 18 - 20 Maret
- 1996 **Training Penggunaan Aplikasi Internet** untuk staf PT. Krakatau Steel, Cilegon, 29-31 Mei
- 1996 **Training UNIX Network Administrator** untuk administrator jaringan PT. Krakatau Steel, Bandung, 25-28 Maret
- 1996 **Training jaringan Komputer TCP/IP dan Paket Radio** untuk administrator jaringan Lembaga Pendidikan Perkebunan (LPP) Yogyakarta, Bandung, 9-10 Januari
- 1995 **In-House Training TCP/IP & UNIX Network Administrator** untuk staf PT.Aplikanusa Lintasarta, Jakarta ,22-24 November
- 1995 **Training publik, Applied Computer Internetworking, UNIX Integration to WAN**, Computer Network Research Group, Hotel Chedi, Bandung 18-20 Juli
- 1995 **Training Singkat jaringan Komputer TCP/IP dan Paket Radio**, Politeknik Universitas Nusa Cendana, Kupang , 26-27 Mei
- 1995 **Training Singkat jaringan Komputer TCP/IP dan Paket Radio**, Politeknik Universitas Nusa Cendana, Kupang , 23-24 Mei
- 1995 **Training publik, Applied Computer Internetworking, Novell Integration to WAN**, Computer Network Research Group, Hotel Chedi, Bandung, 16-18 Mei
- 1995 **Program Magang & Training Network Design & Management for Office** untuk administrator jaringan Perum PERURI, bandung, 5-23 Januari
- 1994 **Kursus Singkat Perancangan jaringan Komputer berbasis Paket Radio** untuk YAE Bogor, Bandung, Juli
- 1994 **Kursus Singkat Perancangan Jaringan Komputer berbasis Novell dan Paket** Radio untuk staf Polytechnique Education Development Center ITB, Bandung, Maret

PUBLIKASI

- 2001 **Apache Web Server**, Onno W. Purbo & Akhmad Daniel Sembiring, PT Elex Media, Jakarta
- 2000 **Sistem Operasi Linux Redhat**, Onno W. Purbo & Akhmad Daniel Sembiring, PT Elex Media, Jakarta
- 2000 **Membangun Web E-Commerce**, Onno W. Purbo & Akhmad Daniel Sembiring, PT Elex Media, Jakarta
- 1998 **Java dan JavaScript**, Onno W. Purbo & Akhmad Daniel Sembiring, PT Elex Media,

Jakarta

1997 **Pemrograman JavaScript**, Onno W. Purbo & Akhmad Daniel Sembiring,
Infokomputer

1997 **Akses Database - Web dengan CGI**, Onno W. Purbo & Akhmad Daniel Sembiring,
Infokomputer