

# Odoo

Weston

Published  
with GitBook



# Table of Contents

---

1. [Introduction](#)
2. [Views](#)
  - i. [search](#)
  - ii. [form with notebook\(tabs\)](#)
  - iii. [link button to a model function\(method\)](#) 自定义一个点击一个button执行某个方法.
  - iv. [show\(detail\) with notebook\(tabs\)](#)
  - v. [add button and bind a action\(添加一个button且绑定action\)](#)
  - vi. [menu](#)
  - vii. [视图定义](#)
  - viii. [分组元素\(GROUPING ELEMENTS\)](#)
  - ix. [数据元素\(DATA ELEMENTS\)](#)
    - i. [Field](#)
    - ii. [Button](#)
    - iii. [Label](#)
    - iv. [New Line](#)
  - x. [视图继承](#)
  - xi. [视图事件](#)
  - xii. [取得缺省值](#)
  - xiii. [Smart Buttons in v8](#)
  - xiv. [cube\\_views](#)
3. [Security](#)
4. [Module](#)
  - i. [structrue](#)
  - ii. [MVC 模式](#)
5. [Achitectrue](#)
  - i. [overview](#)
6. [Create the first Module](#)
7. [Controller](#)
8. [Model](#)
  - i. [domain](#)
9. [Object](#)
  - i. [对象定义的属性](#)
    - i. [必须属性](#)
    - ii. [可选属性](#)
  - ii. [OPENERP 对象字段的定义](#)
  - iii. [字段定义的参数](#)
  - iv. [OpenERP 对象预定义方法](#)
10. [菜单和动作](#)
  - i. [菜单](#)
  - ii. [动作](#)
    - i. [概述](#)
    - ii. [事件例子](#)

- iii. [动作定义](#)
- iv. [window Action](#)
- v. [example of actions](#)
- vi. [Wizard Action](#)
- vii. [Report Action](#)
- viii. [Url Action](#)

## 11. [workflows](#)

- i.  [workflows定义](#)
- ii. [活动定义](#)
- iii. [迁移定义](#)
- iv. [example](#)

## 12. [报表开发](#)

- i. [报表运行机制](#)
- ii. [RML报表开发方法](#)
  - i. [RML语法格式](#)
  - ii. [report action 配置](#)
  - iii. [Parse](#)
- iii. [Aeroo 报表开发方法](#)
  - i. [概述](#)
  - ii. [Aeroo 语法](#)
  - iii. [Aeroo 配置](#)
  - iv. [Parse](#)

## 13. [权限设置](#)

## 14. [Web 开发](#)

- i. [web 运行机制](#)
- ii. [web 页面开发](#)

## 15. [From 7 to 8](#)

- i. [dev tutorial](#)

## Odoo Books

---

Welcome in my book!

参考：

<http://www.oscg.cn:8069/web/static/test/oscgdocs/html/develops/website.html>

<http://www.withfan.com/newest/>

<https://github.com/gxbsst/odoodays-2014>

When to use api.one and api.multi in odoo | openerp?

<http://stackoverflow.com/questions/27988565/when-to-use-api-one-and-api-multi-in-odoo-openerp>

How to override create,write,unlink method in Odoo v8

<http://www.cnblogs.com/kfx2007/p/4134544.html>

## Views

---

## search

Equipments

创建

或 导入

(1-1) 总 1

过滤器

Contracted OwnStatic IP DHCPWith BackupWithout BackupWith AccessWithout AccessWith ApplicationsWithout ApplicationsBundle TypeVirtual TypeProduct TypePartitionedDomain ControllersFileServersHostsRoutersDatabase ServersVPN ServersFirewall and ProxyDHCP ServersAccess PointRegistered by my

保存当前过滤器

高级搜索

添加到控制面板

Group By...

Partner TypeHostContractorRegistrant

| <input type="checkbox"/> | Name      | Description | Equipment Type | Functions |
|--------------------------|-----------|-------------|----------------|-----------|
| <input type="checkbox"/> | 萨法 - 阿萨德发 | 111         | BUNDLE         | (0 条记录)   |
|                          |           |             |                |           |
|                          |           |             |                |           |
|                          |           |             |                |           |
|                          |           |             |                |           |

```

<record model="ir.ui.view" id="it_equipment_search" >
    <field name="name">it.equipment.search</field>
    <field name="model">it.equipment</field>
    <field name="arch" type="xml">
        <search string="Equipment">
            <!--Searchs-->
            <field name="identification" string="Identification Name"/>
            <field name="pin" string="PIN"/>
            <field name="partner_id" string="Partner"/>
            <field name="equipment_type" string="Type"/>
            <field name="virtual_parent_id" string="Host"/>
            <!--Filters-->
            <filter name="is_contracted" string="Contracted" domain="[('is_contra
            <filter name="is_contracted" string="Own" domain="[('is_contracted','
            <separator/>
            <filter name="is_static_ip" string="Static IP" domain="[('is_static_i
            <filter name="is_static_ip" string="DHCP" domain="[('is_static_ip','=
            <separator/>
            <filter name="is_backup" string="With Backup" domain="[('is_backup','
                help="Equipments with backup"/>
            <filter name="is_backup" string="Without Backup" domain="[('is_backup
                help="Equipments without backup"/>
            <separator/>
            <filter name="is_access" string="With Access" domain="[('is_access','
                help="Equipments with access"/>
            <filter name="is_access" string="Without Access" domain="[('is_access
                help="Equipments without access"/>
            <separator/>
            <filter name="is_application" string="With Applications" domain="[('i
                help="Equipments with applications"/>
            <filter name="is_application" string="Without Applications" domain="[
                help="Equipments without applications"/>
            <separator/>
            <filter string="Bundle Type" domain="[('equipment_type','=', 'bundle')
            <filter string="Virtual Type" domain="[('equipment_type','=', 'virtual
            <filter string="Product Type" domain="[('equipment_type','=', 'product
            <separator string="Functions"/>
            <filter name="is_partitioned" string="Partitioned" domain="[('is_part
                help="Equipments with partitions"/>
            <filter name="function_dc" string="Domain Controllers" domain="[('fun
            <filter name="function_fileserver" string="FileServers" domain="[('fu

```

```

<filter name="function_host" string="Hosts" domain="['function_host'
<filter name="function_router" string="Routers" domain="['function_r
<filter name="function_database" string="Database Servers" domain="['
<filter name="function_vpn" string="VPN Servers" domain="['function_
<filter name="function_firewall" string="Firewall and Proxy" domain="
<filter name="function_dhcp" string="DHCP Servers" domain="['function_
<filter name="function_ap" string="Access Point" domain="['function_
<separator/>
<filter domain="['user_id','=',uid)]" help="Registered by my"/>
<!--Groups-->
<group expand="0" string="Group By...">
    <filter string="Partner" icon="terp-personal" domain="[]" context=
    <filter string="Type" icon="terp-personal" domain="[]" context="{
    <filter string="Host" icon="terp-personal" domain="[]" context="{
    <filter string="Contractor" icon="terp-personal" domain="[]" cont
    <filter string="Registrant" icon="terp-personal" domain="[]" cont
</group>
</search>
</field>
</record>

```

## form with notebook(tabs)

Name

Equipment

Partner

Write a note..

Information

SSL

Audit Data

Configuration

User

Application

Port

Description

Link

Password

Generate

Encrypted

Encrypt

Information

SSL

Audit Data

Configuration

CSR

选择

另存为

清除

Cert

选择

另存为

清除

Public Key

选择

另存为

清除

Private Key

选择

另存为

清除

Information

SSL

Audit Data

Configuration

Created by

Administrator

Creation Date

2015/03/06

```

<record model="ir.ui.view" id="it_access_form">
    <field name="name">it.access.form</field>
    <field name="model">it.access</field>
    <field name="arch" type="xml">
        <form string="Access">
            <sheet>
                <div class="oe_title">
                    <div class="oe_edit_only">
                        <label for="name"/>
                    </div>
                    <h1>
                        <field name="name"/>
                    </h1>
                    <div class="oe_edit_only">
                        <label for="equipment_id"/>
                    </div>
                </div>
            </sheet>
        </form>
    </field>
</record>

```



```

        <h1>
            <field name="equipment_id"/>
        </h1>
        <div class="oe_edit_only">
            <label for="partner_id"/>
        </div>
        <h1>
            <field name="partner_id"/>
        </h1>
    </div>
</group>
<group>
    <!--fields-->
</group>
<field colspan="4" name="note" nolabel="1" placeholder="Write a n
<notebook>
    <page string="Information">
        <group col="4">
            <field name="user"/>
            <field name="application"/>
            <field name="port"/>
            <field name="description"/>
            <field name="link" widget="url" />
        </group>
        <newline/>
        <group col="5">
            <field name="password" on_change="onchange_password(e
            <button name="get_random_password" string="Generate"
                confirm="Are you sure?" colspan="1" icon="gtk
            <button name="encrypt_password" type="object" string=
                icon="gtk-execute" colspan="1" attrs="{ 'invi
            <button name="decrypt_password" type="object" string=
                colspan="1" attrs="{ 'invisible': [('encrypted
            <newline/>
            <field name="encrypted" readonly="True"/>
        </group>
    </page>
    <page string="SSL">
        <group label="SSL">
            <field name="ssl_csr" filename="ssl_csr_filename"/>
            <field name="ssl_csr_filename" invisible="1"/>
            <field name="ssl_cert" filename="ssl_cert_filename"/>
            <field name="ssl_cert_filename" invisible="1"/>
            <field name="ssl_publickey" filename="ssl_publickey_f
            <field name="ssl_publickey_filename" invisible="1"/>
            <field name="ssl_privatekey" filename="ssl_privatekey
            <field name="ssl_privatekey_filename" invisible="1"/>
        </group>
    </page>
    <page string="Audit Data" groups="it.group_it_mod">
        <group>
            <group>
                <field name="user_id"/>
                <field name="creation_date" />
            </group>
        </group>
    </page>
    <page string="Configuration" groups="it.group_it_mod">
        <group>
            <group>

```

```

        <field name="active" />
        <field name="company_id" string="Company" groups=
    </group>
    </group>
    </page>
</notebook>
</sheet>
</form>
</field>
</record>

```

**link button to a model function(method)** 自定义一个点击一个button执行某个方法.

The screenshot shows an Odoo form with the following structure:

- Name**: Text input field.
- Equipment**: Dropdown menu.
- Partner**: Dropdown menu.
- Write a note..**: Text area.
- Tabs**: Information (selected), SSL, Audit Data, Configuration.
- Information Tab Fields**:
  - User**: Text input.
  - Port**: Text input.
  - Link**: Text input.
  - Password**: Text input.
  - Encrypted**: Checkbox.
  - Application**: Text input.
  - Description**: Text input.
- Buttons**: 'Generate' and 'Encrypt' buttons at the bottom right.

Red arrows point from the text '调用model的方法' to the 'Generate' and 'Encrypt' buttons.

file: access\_view.xml

```
<record model="ir.ui.view" id="it_access_form">
    <field name="name">it.access.form</field>
    <field name="model">it.access</field>
    <field name="arch" type="xml">
        <form string="Access">
            <sheet>
                <div class="oe_title">
                    <div class="oe_edit_only">
                        <label for="name"/>
                    </div>
                    <h1>
                        <field name="name"/>
                    </h1>
                    <div class="oe_edit_only">
                        <label for="equipment_id"/>
                    </div>
                    <h1>
                        <field name="equipment_id"/>
                    </h1>
                    <div class="oe_edit_only">
                        <label for="partner_id"/>
                    </div>
                    <h1>
                        <field name="partner_id"/>
                    </h1>
                </div>
            </sheet>
        </form>
    </field>
</record>
```

link button to a model function(method) 自定义一个点击一个button执行某个方法.

```

</div>
<group>
  <!--fields-->
</group>
<field colspan="4" name="note" nolabel="1" placeholder="Write a n
<notebook>
  <page string="Information">
    <group col="4">
      <field name="user"/>
      <field name="application"/>
      <field name="port"/>
      <field name="description"/>
      <field name="link" widget="url" />
    </group>
    <newline/>
    <group col="5">
      <field name="password" on_change="onchange_password(e
      <button name="get_random_password" string="Generate"
        confirm="Are you sure?" colspan="1" icon="gtk
      <button name="encrypt_password" type="object" string=
        icon="gtk-execute" colspan="1" attrs="{ 'invi
      <button name="decrypt_password" type="object" string=
        colspan="1" attrs="{ 'invisible': [('encrypted
      <newline/>
      <field name="encrypted" readonly="True"/>
    </group>
  </page>
  <page string="SSL">
    <group label="SSL">
      <field name="ssl_csr" filename="ssl_csr_filename"/>
      <field name="ssl_csr_filename" invisible="1"/>
      <field name="ssl_cert" filename="ssl_cert_filename"/>
      <field name="ssl_cert_filename" invisible="1"/>
      <field name="ssl_publickey" filename="ssl_publickey_f
      <field name="ssl_publickey_filename" invisible="1"/>
      <field name="ssl_privatekey" filename="ssl_privatekey
      <field name="ssl_privatekey_filename" invisible="1"/>
    </group>
  </page>
  <page string="Audit Data" groups="it.group_it_mod">
    <group>
      <group>
        <field name="user_id"/>
        <field name="creation_date" />
      </group>
    </group>
  </page>
  <page string="Configuration" groups="it.group_it_mod">
    <group>
      <group>
        <field name="active" />
        <field name="company_id" string="Company" groups=
      </group>
    </group>
  </page>
</notebook>
</sheet>
</form>
</field>

```

</record>

file: access.py

```
def get_random_password(self, cr, uid, ids, context=None):
    logging.info(".....1")
    for access in self.browse(cr, uid, ids, context=context):
        longitud = 16
        valores = "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ<=>@"
        p = ""
        p = p.join([choice(valores) for i in range(longitud)])
        # self.write(cr, uid, ids, {'password': p, 'encrypted': False })
    return True

def onchange_password(self, cr, uid, ids, encrypted, context={}):
    return {'value':{'encrypted': False}}

def encrypt_password(self, cr, uid, ids, *args):
    logging.info('.....')
    try:
        from Crypto.Cipher import ARC4
    except ImportError:
        raise osv.except_osv(_('Error !'), _('Package python-crypto no installed.))

    val = self.pool.get('ir.config_parameter').get_param(cr, uid, 'it_passkey')
    if not val:
        raise osv.except_osv(_('Error !'), _('For Encryption you must set a system

for rec in self.browse(cr, uid, ids):

    if not rec.encrypted:
        enc = ARC4.new(val)
        try:
            encrypted = base64.b64encode(enc.encrypt(rec.password))
        except UnicodeEncodeError:
            break
        self.write(cr, uid, [rec.id], {'password': encrypted, 'encrypted': True})
    else:
        raise osv.except_osv(_('Error !'), _('Password already encrypted'))

    return True

def decrypt_password(self, cr, uid, ids, *args):
    try:
        from Crypto.Cipher import ARC4
    except ImportError:
        raise osv.except_osv(_('Error !'), _('Package python-crypto no installed.))

    val = self.pool.get('ir.config_parameter').get_param(cr, uid, 'it_passkey')
    if not val:
        raise osv.except_osv(_('Error !'), _('For Encryption you must set a system

for rec in self.browse(cr, uid, ids):
    dec = ARC4.new(val)
```

```


try:
    desencrypted = dec.decrypt(base64.b64decode(rec.password))
    unicode(desencrypted, 'ascii')
    raise osv.except_osv(_('Decrypt password:'), desencrypted)
except UnicodeDecodeError:
    raise osv.except_osv(_('Error !'), _('Wrong encrypt/decrypt password.'))

return True

```

show(detail) with notebook(tabs)

)



萨法

111

Agrolait, 阿萨德发

Physical Information

Worklog

Audit Data

Configuration

| Short Description | Date | Spent Time | User |
|-------------------|------|------------|------|
|                   |      |            |      |
|                   |      |            |      |
|                   |      |            |      |
|                   |      |            |      |



萨法

111

Agrolait, 阿萨德发

Physical Information

Worklog

Audit Data

Configuration

Equipment Type

Equipment Type

BUNDLE

Basic Configuration

Function Configuration

Active

Access

Backup

Applications

Contracted Service

Static IP

Partitions

Store Config Files

Operating System

☒

☐

☐

☐

☐

☐

☐

☐

☐

Host

Router

Domain Controller

File Server

Database Server

VPN Server

Firewall & Proxy Server

DHCP Server

Access Point

☐

☐

☐

☐

☐

☐

☐

☐

☐

```
<record model="ir.ui.view" id="it_equipment_form">
  <field name="name">it.equipment.form</field>
```

```

<field name="model">it.equipment</field>
<field name="arch" type="xml">
  <form string="Equipments">
    <sheet>
      <field name="image" class="oe_left oe_avatar" widget="image"/>
      <div class="oe_title">
        <div class="oe_edit_only oe_left">
          <label for="name"/>
        </div>
        <h1>
          <field name="name"/>
        </h1>
        <div class="oe_edit_only">
          <label for="description"/>
        </div>
        <h1>
          <field name="description"/>
        </h1>
        <div class="oe_edit_only">
          <label for="partner_id"/>
        </div>
        <h1>
          <field name="partner_id"/>
        </h1>
        <div class="oe_edit_only">
          <label for="function_ids"/>
        </div>
        <h1>
          <field name="function_ids" widget="many2many_tags" placeh
        </h1>
      </div>
      <div class="oe_right oe_button_box" name="buttons">
        <button name="%(act_equipment_2_access)d"
          class="oe_inline oe_stat_button"
          icon="fa-key"
          type="action"
          attrs="{ 'invisible': [('is_access', '=', False )]}">
          <field name="access_count" widget="statinfo"/>
        </button>
        <button name="%(act_equipment_2_equipment)d"
          class="oe_inline oe_stat_button"
          icon="fa-th"
          type="action"
          attrs="{ 'invisible': [('function_host', '=', False )]}">
          <field name="virtual_count" widget="statinfo"/>
        </button>
        <button name="%(act_equipment_2_backup)d"
          class="oe_inline oe_stat_button"
          icon="fa-floppy-o"
          type="action"
          attrs="{ 'invisible': [('is_backup', '=', False )]}">
          <field name="backup_count" widget="statinfo"/>
        </button>
      </div>
    <separator/>
    <group>
      <!-- groups and fields
    <group string="Type">

```



```

        </group>
        -->
    </group>
    <field colspan="4" name="note" nolabel="1" placeholder="Write a n
    <notebook>
        <page string="Physical Information" attrs="{ 'invisible': [('eq
            <group>
                <field name="physical_component_ids" nolabel="1" cols
            </group>
        </page>
        <page string="Virtual Machine Data" attrs="{ 'invisible': [('eq
            <group>
                <group string="Host">
                    <field name="virtual_parent_id"/>
                </group>
                <group string="Configuration">
                    <field name="virtual_memory_amount"/>
                    <field name="virtual_disk_amount"/>
                    <field name="virtual_processor_amount"/>
                    <field name="virtual_network_amount"/>
                </group>
            </group>
        </page>
        <page string="Product Data" attrs="{ 'invisible': [('equipment_
            <group>
                <group>
                    <field name="product_id"/>
                    <field name="product_serial_number"/>
                    <field name="product_warranty"/>
                    <field name="product_buydate"/>
                </group>
                <group>
                    <field name="product_note"/>
                </group>
            </group>
        </page>
        <page string="Partition Information" attrs="{ 'invisible': [('i
            <group>
                <field name="partitions_ids" nolabel="1" colspan="4"/
            </group>
        </page>
        <page string="Service Information" attrs="{ 'invisible': [('is_
            <group>
                <group string="Contract Information">
                    <field name="contract_partner_id"/>
                    <field name="contract_client_number"/>
                    <field name="contract_owner"/>
                    <field name="contract_nif"/>
                    <field name="contract_direction"/>
                </group>
            </group>
        </page>
        <page string="Router Configuration" attrs="{ 'invisible': [('fu
            <group>
                <field name="router_dmz"/>
                <field name="router_forward_ids">
                    <tree string="Forward Registry" editable="bottom"
                        <field name="name"/>
                        <field name="source_port"/>

```

```

        <field name="destination_port"/>
        <field name="source_address"/>
        <field name="destination_address"/>
        <field name="type"/>
    </tree>
</field>
<field name="router_nat_ids">
    <tree string="Nat Registry" editable="bottom">
        <field name="name"/>
        <field name="source_address"/>
        <field name="destination_address"/>
    </tree>
</field>
<field name="router_rules_ids">
    <tree string="Rules Registry" editable="bottom">
        <field name="name"/>
        <field name="source_port"/>
        <field name="destination_port"/>
        <field name="source_address"/>
        <field name="destination_address"/>
        <field name="permission"/>
    </tree>
</field>
</group>
</page>
<page string="Network Configuration" attrs="{ 'invisible': [('i
    <group>
        <group string="Network Configuration">
            <field name="equipment_network_ids" nolabel="1" c
        </group>
    </group>
</page>
<page string="File Server" attrs="{ 'invisible': [('function_fi
    <group>
        <field name="equipment_mapping_ids" nolabel="1" colsp
    </group>
</page>
<page string="Domain Controller" attrs="{ 'invisible': [('funct
    <group>
        <field name="dc_name"/>
        <field name="dc_type"/>
        <field name="dc_user_ids">
            <tree string="Users" editable="bottom">
                <field name="name"/>
                <field name="user"/>
                <field name="mail"/>
                <field name="actived"/>
            </tree>
        </field>
        <field name="dc_group_ids">
            <tree string="Groups" editable="bottom">
                <field name="name"/>
                <field name="description"/>
                <field name="actived"/>
            </tree>
        </field>
    </group>
</page>
<page string="Wireless Config" attrs="{ 'invisible': [('functio

```

```

<group string="Wifi">
  <field name="ap_ssid"/>
  <field name="ap_auth_type"/>
  <field name="ap_password"/>
</group>
<group string="Guest Access">
  <field name="ap_guest"/>
  <field name="ap_guest_ssid" attrs="{ 'invisible': [('ap
  <field name="ap_guest_password" attrs="{ 'invisible': [
</group>
</page>
<page string="DHCP Config" attrs="{ 'invisible': [('function_dh
  <group>
    <field name="dhcp_scope"/>
    <field name="dhcp_relay"/>
    <field name="dhcp_reservation_ids">
      <tree string="Reservations" editable="bottom">
        <field name="name"/>
        <field name="mac_address"/>
        <field name="ip_address"/>
      </tree>
    </field>
  </group>
</page>
<page string="Databases" attrs="{ 'invisible': [('function_data
  <group>
    <field name="db_engine"/>
    <field name="db_setting_ids">
      <tree string="Engine Settings" editable="bottom">
        <field name="name"/>
        <field name="value"/>
        <field name="description"/>
      </tree>
    </field>
    <field name="db_ids">
      <tree string="Databases" editable="bottom">
        <field name="name"/>
        <field name="codification"/>
        <field name="description"/>
      </tree>
    </field>
  </group>
</page>
<page string="Firewall and Proxy" attrs="{ 'invisible': [('func
  <group string="Firewall Filter">
    <field name="firewall_filter_ids" nolabel="1" colspan
    <tree string="Firewall Filters" editable="bottom">
      <field name="name"/>
      <field name="source_address"/>
      <field name="destination_address"/>
      <field name="service_port"/>
      <field name="permission"/>
    </tree>
  </field>
</group>
<group string="Proxy">
  <field name="proxy_transparent"/>
  <field name="proxy_enable_ssk"/>
  <field name="proxy_adblocking"/>

```

```

        <field name="proxy_port"/>
        <field name="proxy_cache_size"/>
    </group>
</page>
<page string="VPN" attrs="{ 'invisible': [('function_vpn', '=',
    <group string="VPN Server Config">
        <field name="vpn_protocol"/>
        <field name="vpn_address"/>
        <field name="vpn_server_cert"/>
        <field name="vpn_cacn"/>
        <field name="vpn_tun"/>
        <field name="vpn_nat"/>
        <field name="vpn_c2c"/>
        <field name="vpn_gateway"/>
        <field name="vpn_search_domain"/>
        <field name="vpn_wins_server"/>
    </group>
</page>
<page string="Operating System" attrs="{ 'invisible': [('is_os'
    <group>
        <field name="os_name"/>
        <field name="os_company"/>
        <field name="os_version"/>
    </group>
</page>
<page string="Applications" attrs="{ 'invisible': [('is_applica
    <group>
        <field name="application_ids" nolabel="1" colspan="4"
    </group>
</page>
<page string="Configuration Files" attrs="{ 'invisible': [('is_
    <group>
        <field name="configuration_file_ids">
            <tree string="Configuration Files" editable="bott
                <field name="name"/>
                <field name="date"/>
                <field name="config_file" filename="config_fi
                <field name="config_file_filename" invisible=
            </tree>
        </field>
    </group>
</page>
<page string="Worklog" >
    <field name="worklog_ids" nolabel="1" colspan="4"/>
</page>
<page string="Audit Data" groups="it.group_it_mod">
    <group>
        <group>
            <field name="user_id"/>
            <field name="creation_date"/>
            <field name="pin"/>
        </group>
    </group>
</page>
<page string="Configuration" groups="it.group_it_mod">
    <group string="Equipment Type">
        <field name="equipment_type"/>
    </group>
<newline/>

```

```

<group>
  <group string="Basic Configuration">
    <field name="active"/>
    <field name="company_id" string="Company" groups=
    <field name="is_access"/>
    <field name="is_backup"/>
    <field name="is_application"/>
    <field name="is_contracted"/>
    <field name="is_static_ip"/>
    <field name="is_partitioned"/>
    <field name="is_config_file"/>
    <field name="is_os"/>
  </group>
  <group string="Function Configuration">
    <field name="function_host"/>
    <field name="function_router"/>
    <field name="function_dc"/>
    <field name="function_fileserver"/>
    <field name="function_database"/>
    <field name="function_vpn"/>
    <field name="function_firewall"/>
    <field name="function_dhcp"/>
    <field name="function_ap"/>
  </group>
</group>
</page>
</notebook>
</sheet>
</form>
</field>
</record>

```

## add button and bind a action



file: equipment\_view.xml

### 1. 定义action

```
<record id="act_equipment_2_access" model="ir.actions.act_window">
  <field name="res_model">it.access</field>
  <field name="view_type">form</field>
  <field name="name">Access</field>
  <field name="view_mode">tree, form</field>
  <field name="context">{'search_default_equipment_id': active_id }</field>
</record>
```

### 1. 定义button且绑定action

```
<div class="oe_right oe_button_box" name="buttons">
  <button name="%(act_equipment_2_access)d"
    class="oe_inline oe_stat_button"
    icon="fa-key"
    type="action"
    attrs="{ 'invisible': [('is_access', '=', True)] }">
    <field name="access_count" widget="statinfo"/>
  </button>
  <button name="%(act_equipment_2_equipment)d"
    class="oe_inline oe_stat_button"
    icon="fa-th"
    type="action"
    attrs="{ 'invisible': [('function_host', '=', True)] }">
    <field name="virtual_count" widget="statinfo"/>
  </button>
  <button name="%(act_equipment_2_backup)d"
    class="oe_inline oe_stat_button"
    icon="fa-floppy-o"
    type="action"/>
```

```
        type="action"
        attrs="{ 'invisible':[('is_backup','=', True )]}"
        <field name="backup_count" widget="statinfo"/>
    </button>
</div>
```

## 参考

---

这个文章有非常的参考价值，关于 **smart button**

<http://www.slideshare.net/openobject/odoo-smart-buttons>

## menu

---

```
<!-- Top menu item -->
<menuitem name="Infrastructure" id="it.menu_main_itm" groups="group_it_user" sequ

<!-- General Menu -->
<menuitem name="General Menu" id="it_menu_general" parent="it.menu_main_itm" grou

<!-- Others -->
<menuitem name="Other" id="it_menu_others" parent="it.menu_main_itm" groups="grou
```



## 视图定义

---

### 示例

```
<?xml version="1.0"?>
<openerp>
<data>
[view definitions]
</data>
</openerp>
```

The view definitions contain mainly three types of tags: • tags with the attribute `model="ir.ui.view"`, which contain the view definitions themselves • tags with the attribute `model="ir.actions.act_window"`, which link actions to these views • tags, which create entries in the menu, and link them with actions

New : You can precise groups for whom the menu is accessible using the `groups` attribute in `menuitem` tag. New : You can now add shortcut using the `shortcut` tag. Example

Note that you should add an `id` attribute on the `menuitem` which is referred by `menu` attribute.

sale.order.form sale.order

.....

Default value for the `priority` field : 16. When not specified the system will use the view with the lower priority.

## grouping elements

---


Separator 视图上增加一条分割线,例:

The string attribute defines its label and the colspan attribute defines his horizontal size (in number of columns). Notebook

: 定义Tab页。With notebooks you can distribute the view fields on different tabs (each one defined by a page tag). You can use the tabpos properties to set tab at: up, down, left, right. 示例:

.... Group

: groups several columns and split the group in as many columns as desired. • colspan: the number of columns to use • rowspan: the number of rows to use • expand: if we should expand the group or not • col: the number of columns to provide (to its children) • string: (optional) If set, a frame will be drawn around the group of fields, with a label containing the string. Otherwise, the frame will be invisible.

Example ``` python 

Page Defines a new notebook page for the view. Example

... : • string: defines the name of the page.

# 数据元素(DATA ELEMENTS)

---

# Field

## Field

attributes for the “field” tag

- `select="1"`: mark this field as being one of the research criteria for this resource search view.
- `colspan="4"`: the number of columns on which a field must extend.
- `readonly="1"`: set the widget as read only
- `required="1"`: the field is marked as required. If a field is marked as required, a user has to fill it the system won't save the resource if the field is not filled. This attribute supersede the required field value defined in the object.
- `no_label="1"`: hides the label of the field (but the field is not hidden in the search view).
- `invisible="True"`: hides both the label and the field.
- `string=""`: change the field label. Note that this label is also used in the search view: see select attribute above).
- `domain`: can restrict the domain. – Example: `domain="[('partner_id','=',partner_id)]"`
- `widget`: can change the widget. – Example: `widget="one2many_list" * one2one_list`
- `one2many_list`
- `many2one_list`
- `many2many`
- `url`
- `email`
- `image`
- `float_time`
- `reference`
- `on_change`: define a function that is called when the content of the field changes. – Example: `on_change="onchange_partner(type,partner_id)"` – See `ViewsSpecialProperties` for details
- `attrs`: Permits to define attributes of a field depends on other fields of the same window. (It can be use on page, group, button Format: “{'attribute':[(‘field\_name’,‘operator’,‘value’), (‘field\_name’,‘operator’,‘value’)],‘attribute2’:[(‘field\_name’,‘operator’,’ – where attribute will be readonly, invisible, required – Default value: {}.

– 示例: (in `product.product`)

```
<field digits="(14, 3)" name="volume" attrs="{ 'readonly': [( 'type', '=', 'service' )] }"/>
```

示例 Here's the source code of the view of a sale order object. This is the same object as the object shown on the screen

```
<?xml version="1.0"?>
<terp>
<data>
<record id="view_partner_form" model="ir.ui.view">
<field name="name">res.partner.form</field> <field name="model">res.partner</field> <field
<field name="arch" type="xml">
<form string="Partners"> <group colspan="4" col="6">
<field name="name" select="1"/> <field name="ref" select="1"/>

<field name="customer" select="1"/>
<field domain="[('domain', '=', 'partner')]" name="title"/> <field name="lang" select="2"
<field name="supplier" select="2"/>
</group>
<notebook colspan="4"> <page string="General">
<field colspan="4" mode="form,tree" name="address" nolabel="1" select="1"> <form string="
<field name="name" select="2"/>
<field domain="[('domain', '=', 'contact')]" name="title"/>
<field name="function"/>
<field name="type" select="2"/>
<field name="street" select="2"/>
<field name="street2"/>
<newline/>
<field name="zip" select="2"/>
<field name="city" select="2"/>
<newline/>
<field completion="1" name="country_id" select="2"/> <field name="state_id" select="2"/>
<newline/>
<field name="phone"/>
<field name="fax"/>
<newline/>
<field name="mobile"/>
<field name="email" select="2" widget="email"/>
</form>
<tree string="Partner Contacts">
<field name="name"/> <field name="zip"/>
<field name="city"/>
<field name="country_id"/> <field name="phone"/> <field name="email"/>
</tree> </field>
<separator colspan="4" string="Categories"/>
<field colspan="4" name="category_id" nolabel="1" select="2"/> </page>
<page string="Sales & Purchases">
<separator string="General Information" colspan="4"/> <field name="user_id" select="2"/>
<field name="active" select="2"/>
<field name="website" widget="url"/>
<field name="date" select="2"/>
<field name="parent_id"/>
<newline/>
</page>
<page string="History">
<field colspan="4" name="events" nolabel="1" widget="one2many_list"/> </page>
```

# Button

## 参考

[https://github.com/odoo/odoodays-2014/blob/master/smart\\_buttons/index.rst](https://github.com/odoo/odoodays-2014/blob/master/smart_buttons/index.rst)

: add a button using the string attribute as label. When clicked, it can trigger methods on the object, workflow transitions or actions (reports, wizards, ...).

- string: define the button's label
- confirm: the message for the confirmation window, if needed. Eg: confirm="Are you sure?"
- name: the name of the function to call when the button is pressed. In the case it's an object function, it must take 4 arguments:
  - cr is a database cursor
  - uid is the userID of the user who clicked the button
  - ids is the record ID list
  - \*\*args is a tuple of additional arguments
- states: a comma-separated list of states (from the state field or from the workflow) in which the button must appear. If the states attribute is not given, the button is always visible.
- type: this attribute can have 3 values – "workflow" (value by default): the function to call is a function of workflow
  - "object": the function to call is a method of the object
  - "action": call an action instead of a function

## Example

```
<button name="order_confirm" states="draft" string="Confirm Order" icon="gtk-execute"
```

## Label

---

Adds a simple label using the string attribute as caption. Example

```
<label string="Test"/>
```

## New Line

---

Force a return to the line even if all the columns of the view are not filled in. Example

```
<newline/>
```



## 视图继承

When you create and inherit objects in some custom or specific modules, it is better to inherit (than to replace) from an existing view to add/modify/delete some fields and preserve the others. Example

```
<record model="ir.ui.view" id="view_partner_form">
<field name="name">res.partner.form.inherit</field>
<field name="model">res.partner</field>
<field name="inherit_id" ref="base.view_partner_form"/>
<field name="arch" type="xml">
<notebook position="inside">
<page string="Relations">
<field name="relation_ids" colspan="4" nolabel="1"/> </page>
</notebook>
</field>
</record>
```

The inheritance engine will parse the existing view and search for the the root nodes of

```
<field name="arch" type="xml">
```

It will append or edit the content of this tag. If this tag has some attributes, it will look for the matching node, including the same attributes (unless position). This will add a page to the notebook of the res.partner.form view in the base module. You can use these values in the position attribute:

- inside (default): your values will be appended inside this tag
- after: add the content after this tag
- before: add the content before this tag
- replace: replace the content of the tag.

### Second Example

```
<record model="ir.ui.view" id="view_partner_form"> <field name="name">res.partner.form.in
<field name="model">res.partner</field>
<field name="inherit_id" ref="base.view_partner_form"/>
<field name="arch" type="xml">
<page string="Extra Info" position="replace">
<field name="relation_ids" colspan="4" nolabel="1"/>
</page>
</field>
</record>
```

Will replace the content of the Extra Info tab of the notebook by one 'relation\_ids' field. The parent and the inherited views are correctly updated with –update=all argument like any other views.

To delete a field from a form, an empty element with position="replace" attribute is used. Example:

```
<record model="ir.ui.view" id="view_partner_form3"> <field name="name">res.partner.form.i
<field name="model">res.partner</field>
<field name="inherit_id" ref="base.view_partner_form"/>
<field name="arch" type="xml">
<field name="lang" position="replace"/> </field>
</record>
```

Take into account that only one position="replace" attribute can be used per inherited view so multiple inherited views must be created to make multiple replacements.

## 视图事件

### On Change

The `on_change` attribute defines a method that is called when the content of a view field has changed. This method takes at least arguments: `cr`, `uid`, `ids`, which are the three classical arguments and also the context dictionary. You can add parameters to the method. They must correspond to other fields defined in the view, and must also be defined in the XML with fields defined this way:

```
<field name="name_of_field" on_change="name_of_method(other_field'_1_', ..., other_field'
```

The example below is from the sale order view. You can use the 'context' keyword to access data in the context that can be used as params of the function.:

```
<field name="shop_id" select="1" on_change="onchange_shop_id(shop_id)"/>

def onchange_shop_id(self, cr, uid, ids, shop_id):
    v={}
    if shop_id: shop=self.pool.get('sale.shop').browse(cr,uid,shop_id) v['project_id']=shop.p
    if shop.pricelist_id.id: v['pricelist_id']=shop.pricelist_id.id v['payment_default_id']=s
```

When editing the `shop_id` form field, the `onchange_shop_id` method of the `sale_order` object is called and returns a dictionary where the 'value' key contains a dictionary of the new value to use in the 'project\_id', 'pricelist\_id' and 'payment\_default\_id' fields.

Note that it is possible to change more than just the values of fields. For example, it is possible to change the value of some fields and the domain of other fields by returning a value of the form: `return {'domain': d, 'value': value}` context in `<record model="ir.actions.act_window" id="a">` you can add a context field, which will be pass to the action.

See the example below:

```
<record model="ir.actions.act_window" id="a">
<field name="name">account.account.tree1</field> <field name="res_model">account.account<
<field name="view_mode">form,tree</field> <field name="view_id" ref="v"/>
<field name="domain">[('code','=', '0')]</field>
<field name="context">{'project_id': active_id}</field> </record>
```

#### view\_type:

tree = (tree with shortcuts at the left), form = (switchable view form/list)

**view\_mode** tree,form : sequences of the views when switching

## get default value

---

## Smart Buttons in v8

---

参考：

[https://github.com/odoo/odoodays-2014/blob/master/smart\\_buttons/index.rst](https://github.com/odoo/odoodays-2014/blob/master/smart_buttons/index.rst)

## Smart Buttons in v8

---

Géry Debongnie

### Content

---

1. What are Smart Buttons?
2. From Regular Buttons to Smart Buttons
3. Magic! (not really)
4. Customizing the Look
5. Customizing the Content

## What are Smart Buttons?

---

### Smart Buttons

---

.. image:: images/formview.png

.. nextslide:: :increment:

#### From

.. image:: images/oldstyle.png :align: center :width: 500px

#### To

.. image:: images/buttons.png :align: center

.. nextslide:: :increment:

Two biggest advantages:

- dynamic,
- customizable.

.. image:: images/buttons.png

## From Regular to Smart

---

it's not hard

## Converting a plain button

---

.. image:: images/opps\_old.png :width: 30%

.. image:: images/opps\_new.png :width: 35%

Before:

.. code-block:: xml

```
<button class="oe_inline" type="action"
    string="Opportunities"
    name="..." context="..."/>
```

After:

.. code-block:: xml

```
<button class="oe_inline oe_stat_button" type="action"
    string="Opportunities"
    name="..." context="..."/>
```

.. nextslide:: :increment:

.. image:: images/opps\_icon.png :width: 30% :align: center

Just add 'icon' attribute.

.. code-block:: xml

```
<button class="oe_inline oe_stat_button" type="action"
    string="Opportunities"
    icon="fa-star"
    name="..." context="..."/>
```

Font awesome:

<http://fontawesome.github.io/Font-Awesome/>

## Where is the magic?

---

(next slide)

..

## Button tag can contain anything

---

Before, the button tag was self-closed:

.. code-block:: xml

```
<button/>
```

Now, it can contain literally anything:

.. code-block:: html

```
<button>
    literally anything
</button>
```

The form view parse the button and render anything inside (html/Odoo widgets)

## Example (html)

---

Pure html : Full control on the content

.. code-block:: xml

```
<button class="..." type="..." name="..." context="...">
    <p>Hello <strong>Odoo</strong></p>
</button>
```

Result:

.. image:: images/helloodoo.png :width: 30% :align: center

## Example (html+field)

---

.. code-block:: xml

```
<button class="..." type="..." name="..." icon="fa-star">
    <span><field name="opportunity_count"/> Opportunities</span>
</button>
```

Result:

.. image:: images/ophtml.png :width: 30% :align: center

This is fully dynamic!



## Common situation: One2many fields

---

Example: *phonecall\_ids* in *res.partner*.

Step 1: add functional field *phonecall\_count* to *res.partner*

Step 2: add field with widget 'statinfo'

.. code-block:: xml

```
<button class="..." type="..." name="..." icon="..." context="...">
    <field string="Calls" name="phonecall_count" widget="statinfo"/>
</button>
```

.. image:: images/calls.png :width: 35% :align: center

## Customize your buttons

---

We can customize in two ways:

.. image:: images/customizebutton.png :align: center

## Customizing Content

---

### Case study: Sum of all invoices for a customer

---

1. add functional field

.. code-block:: python

```
'total_invoiced': fields.function(_invoice_total,
                                  string="Total Invoiced",
                                  type='float')
```

.. code-block:: python

```
def _invoice_total(self, cr, uid, ids, field_name, arg, context=None):
    result = {}
    account_invoice_report = self.pool.get('account.invoice.report')
    for partner in self.browse(cr, uid, ids, context=context):
        ...
    return result
```

.. nextslide:: :increment:

1. add field to button

.. code-block:: xml

```
<button type="action" class="oe_stat_button"
    icon="fa-pencil-square-o" name="..." context="..." >
    <field name="total_invoiced" widget="statinfo"/>
</button>
```

1. profit!

.. image:: images/totalinvoiced.png :align: center :width: 40%

## Customizing Look

---

### PercentPie Widget

---

Percentage (integer between 0 and 100)

.. image:: images/percentpie.png :width: 35% :align: center

.. code-block:: xml

```
<button name="..." type="action" class="oe_stat_button">
    <field name="received_ratio"
        string="Received"
        widget="percentpie"/>
</button>
```

### Bar Chart Widget

---

Need to display some kind of trends? Use BarChart Widget!

.. image:: images/barchartwidget.png :width: 35% :align: center

.. code-block:: xml

```
<button name="..." type="action" class="oe_stat_button">
    <field name="opened_daily"
        string="Opened Daily"
        widget="barchart"/>
</button>
```

(see mass\_mailing.py for full details)

## Thank you!

---

## cube\_views

---

参考

[https://github.com/odoo/odoodays-2014/blob/master/cube\\_views/index.rst](https://github.com/odoo/odoodays-2014/blob/master/cube_views/index.rst)

## Business Intelligence

---

Develop cube views for your own objects

.. Géry Debongnie

### Content

---

1. BI/Graph View
2. Technical Overview
3. Preparing Data
4. Displaying cube views
5. Case study: cross models
6. Conclusion

## BI/Graph View

---

### BI/Graph View

---

.. image:: images/biview.png :width: 100%

- measure : can be aggregated (right now, only summed)
- dimension : can be grouped

## A Short History of BI in Odoo

---

- pre 2014: list view + group bys, graph view
- Q1/Q2 2014: graph view rewrite -> pivot table + graphs, lots of backend work
- future: ? we're looking at searchview/BI view integration.

## Technical Overview

---

### Odoo architecture

---

.. image:: images/overview.png :align: center

## Anatomy of BI/Graph View

---

.. image:: images/bistructure.png :align: center

- **pivot table**: keeps the data, calls the ORM
- **graph widget** : user interaction
- **graph view** : client interaction

## BI view xml

---

.. code-block:: xml

```
<record id="..." model="ir.ui.view">
  <field name="name">crm.opportunity.report.graph</field>
  <field name="model">crm.lead.report</field>
  <field name="arch" type="xml">
    <graph string="Leads Analysis" type="pivot" stacked="True">
      <field name="date_deadline" type="row"/>
      <field name="stage_id" type="col"/>
      <field name="planned_revenue" type="measure"/>
    </graph>
  </field>
</record>
```

## BI view API

---

In *graph* tag:

- string: title
- stacked: if bar chart is stacked/not stacked (default=false)
- type: mode (pivot, bar, pie, line) (default=bar)

In *field* tags, *type* attribute:

- row : will be grouped by rows (dimension)
- col : will be grouped by cols (dimension)
- measure : will be aggregated
- if no type, measure by default

## Date/datetime

---

Always want to be different: date/datetime have a special syntax for groupby:

- field\_date:day,
- field\_date:week,

- field\_date:month (default)
- field\_date:quarter,
- field\_date:year

.. code-block:: xml

```
<graph string="Leads Analysis" type="pivot" stacked="True">
  <field name="date_deadline:week" type="row"/>
  <field name="stage_id" type="col"/>
  <field name="planned_revenue" type="measure"/>
</graph>
```

## Graph widget setup

---

Graph widget has two more options:

- *visible\_ui* (true) : indicate if UI is visible
- *heatmap\_mode* ('none') can be set to row/col/both

## Preparing Data

---

### Odoo Model

---

Odoo BI view will read the various fields. Depending on their type, it will use them for

- measures : every fields of type *integer*, *float* (except 'id')
- dimensions :
  - right now: every fields defined in the 'group by' category in the search bar.
  - later: every field that can be grouped by the db

## Where is your data?

---

The data needs to satisfy two conditions:

- be stored in the database (beware of functional fields not stored)
- be accessed from one single odoo model

If yes, you're done. If not, two possibilities:

- can you extend a model? (stored functional fields, relational fields)
- can you create a custom model with a postgres view, to link the various models with the data?

.. .. image:: images/choices2.png .. :align: center .. :width: 100%

---

Bottom line: it needs to be in the DB

## Extending a model

---

WARNING: old API... Do not try this at home!!!

.. code-block:: python

```
class res_partner(osv.osv):
    _name = 'res.partner'
    _inherit = 'res.partner'

    def _total_invoice(self, cr, uid, ids, ...):
        ...
        # [insert here nice looking code to
        # compute the total invoice of a customer]
        ...
        return result

    _columns = {
        'total_invoiced': fields.function(_total_invoice,
            string="Total Invoiced", type='float', store=True)
    }
```

.. note::

second warning: untested code!! this is just a proof of concept  
to illustrate the point

emphasize that store=true is required, otherwise it will not work

.. give code example

## More advanced: Cross model analysis

---

.. image:: images/postgresview.png :align: center

Example: purchase/report/purchase\_report.py

All reporting views use that technique. Warning: bypass the ORM

## Displaying cube views

---

### Edit in live

---

1. go to developer mode
2. edit action, add 'graph',

3. edit views, create 'graph'
4. profit!

Good for testing.

## Adding a BI view with xml

---

Add the desired graph view:

.. code-block:: xml

```
<record id="view_project_task_graph" model="ir.ui.view">
  <field name="name">project.task.graph</field>
  <field name="model">project.task</field>
  <field name="arch" type="xml">
    <graph string="Project Tasks" type="bar">
      <field name="project_id" type="row"/>
      <field name="planned_hours" type="measure"/>
    </graph>
  </field>
</record>
```

## Adding a BI view with xml(2)

---

Add it to the action:

.. code-block:: xml

```
<record id="action_view_task" model="ir.actions.act_window">
  ...
  <field name="view_mode">kanban,tree,form,calendar,gantt,graph</field>
  ...
</record>
```

You can force the correct view:

.. code-block:: xml

```
<field name="view_id" ref="view_project_task_graph"/>
```

## Advanced: client action

---

In js, create a widget and append it to your view:

.. code-block:: javascript

```
this.graph_widget = new openerp.web_graph.Graph(
```

```
        this,  
        some_model,  
        some_domain,  
        options);  
this.graph_widget.appendTo(this.$el);
```

## Future of BI in odoo?

---

## Thank you

---



# Security

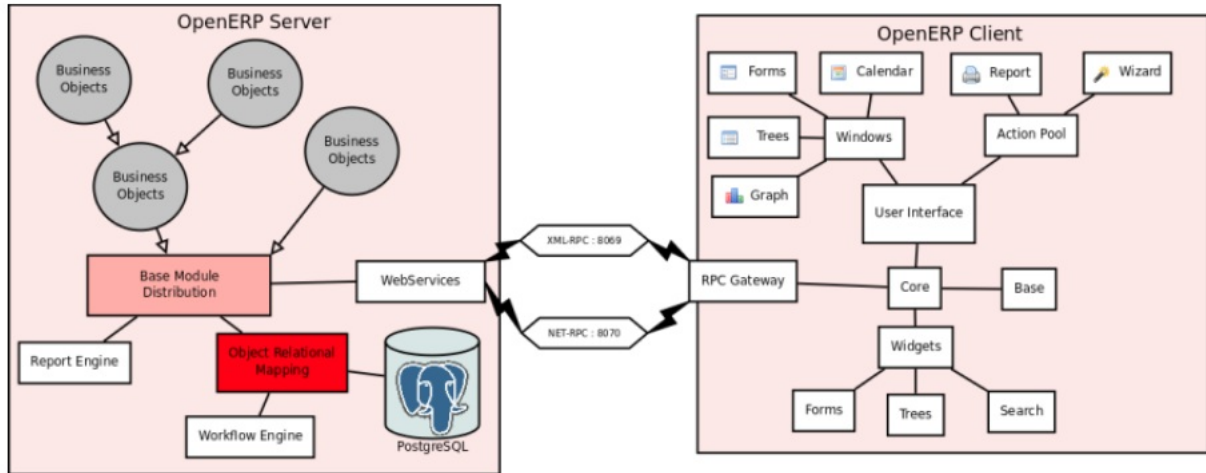
---

# Module

---

## 模块结构

OpenERP 的各种功能都是由一个个的模块提供,OpenERP V7.0 中,系统提供的标准模块有 170 多个,在 OpenERP Apps Store 上面还有近 2000 个模块。OpenERP 的模块,通常由这么一些 对象组成:



- **Business Object:** 负责数据表的增删改查等操作。
- **View:** 将数据表的字段展现在窗口上。
- **Action:** 联系Object和View的Controller。
- **Workflow:** 管理BusinessObject处理流程的工作流。
- **Report:** 将BusinessObject打印出来的报表。
- **Wizard:** 处理一些复杂的用户交互操作功能。 \*

模块典型目录构成如下:

```
module_name/
  __init__.py
  __openerp__.py
  module_name.py
  module_name_2.py
  module_name_3.py
  module_name_view.xml
  module_name_workflow.xml
  module_name_wizard.xml
  module_name_report.xml
  module_name_demo.xml
  module_name_data.xml
report/
  __init__.py
  report.py
wizard/
  __init__.py
  wizard.py
```

```
i18n/
  fr_FR.po
  en_US.po
  module_name.pot
security/
  some.object.csv #定义权限组访问规则
  file.xml       #定义用户组
```

## MVC 模式

当我们操作 OpenERP 的菜单时,通常是点击菜单(如销售 -> 客户),跳出对象选择画面,当选择一条记录时,跳出对象编辑画面。



对象选择画面、对象编辑画面,在 OpenERP 里称为视图(View),选择画面是看板视图 (Kanban View)和列表视图(Tree View),编辑画面是表单视图(Form View)。OpenERP 里的对象(Object),也叫 Model,相当于我们一般说的类(Class),对象总是对应到数据库里的数据表。例如业务伙伴对象,其对象名是“res.partner”,对应表名是“res\_partner”。表里的一条记录,也就是对象的一个实例,叫资源(Resource)。

客户 / Agrolait

保存 or 放弃 1 / 40

名称 ( ☒ 是一个公司? )

Agrolait

会议 通话 商机列表

报价单或销售订单

部件买家 业务伙伴 / IT 服务 Tags...

地址

比利时 State Wavre 区/县 1300 69 rue de Chimay

电话 +32 10 588 558

手机

传真

电子邮件 info@agrolait.com

网站 www.agrolait.com

联系人 内部单据 销售&采购 会计 日志 POS

创建

Michel Fletcher Analyst m.fletcher@agrolait.com

Thomas Passot Functional Consultant p.thomas@agrolait.com

当点击菜单时,系统怎么知道应该跳出哪个画面,以及应该显示哪个对象的记录呢?把菜单和对应的对象、视图关联起来的是 Action。当用户点击菜单时,触发 Action,Action 调用对象的 Search 方法,从数据库取得记录(资源),Action 又创建视图,显示取得的数据。简单总结一下,OpenERP 的开发中,有如下一些重要概念:

- 对象或模型(Object or Model):是一个 Python 的 Class,也对应到数据库的一张数据表,负责存取数据记录(Record),有 Search、Read、Write 等方法。OpenERP 在模块加载时,初始化模块中的所有对象,放入对象池。因此,数据库操作时,通常是先从对象池中取得对象,再调用对象的方法。下面分析一段典型的记录查找和读取代码。

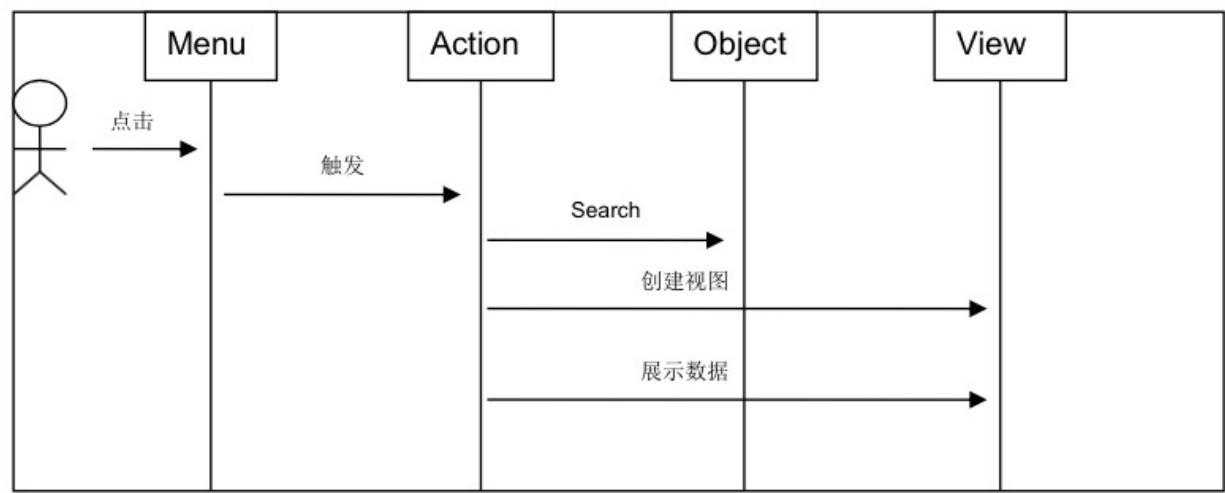
```
def _get_admin_id(self, cr):
    if self.__admin_ids.get(cr.dbname) is None:
        ir_model_data_obj = self.pool.get('ir.model.data')
        mdid = ir_model_data_obj._get_id(cr, 1, 'base', 'user_root')
        self.__admin_ids[cr.dbname] = ir_model_data_obj.read(cr, 1, [mdid], ['res_id'])[0]['res_id']
    return self.__admin_ids[cr.dbname]
```

这段代码来自文件 server\addons\base\res\res\_user.py,它先从对象池取得对象'ir.model.data',该对象负责存取数据表 ir\_model\_data。而后调用该对象的方法\_get\_id 取得 admin 用户的 user\_id,再读入 admin 对应的 user 记录的 id。

- 视图(View):负责显示数据,最常见的视图是列表视图和表单视图。此外,还有看板、日历、甘特图、图形、流程图等几种视图,不同的视图以不同的方式展示数据。本章主要介绍列表和表单,另外几种视图以后介绍。
- 菜单(Menu):这个很直观,不用介绍了。
- 动作(Action):用户操作系统时(如点击菜单、点击画面右边的工具条上的按钮等),系统的响应动作。一个 Action 包含一个对象,包含若干个视图,通常每个 Action 都包含列表和表单两个视图。当 Action 被触

发时,相应的视图被调出,展示相应的对象的数据。Action 有多种类型,最 常见的是 Act\_Window(窗口类型),窗口类型跳出一个窗口以显示数据。此外还有 Report(报 表)、Wizard(向导)等类型。本章主要介绍窗口类型。

上述概念间的关联关系,参见如下操作序列图:



例如,当打开一个财务凭证时(对象 account.invoice),客户端发生的动作链是:

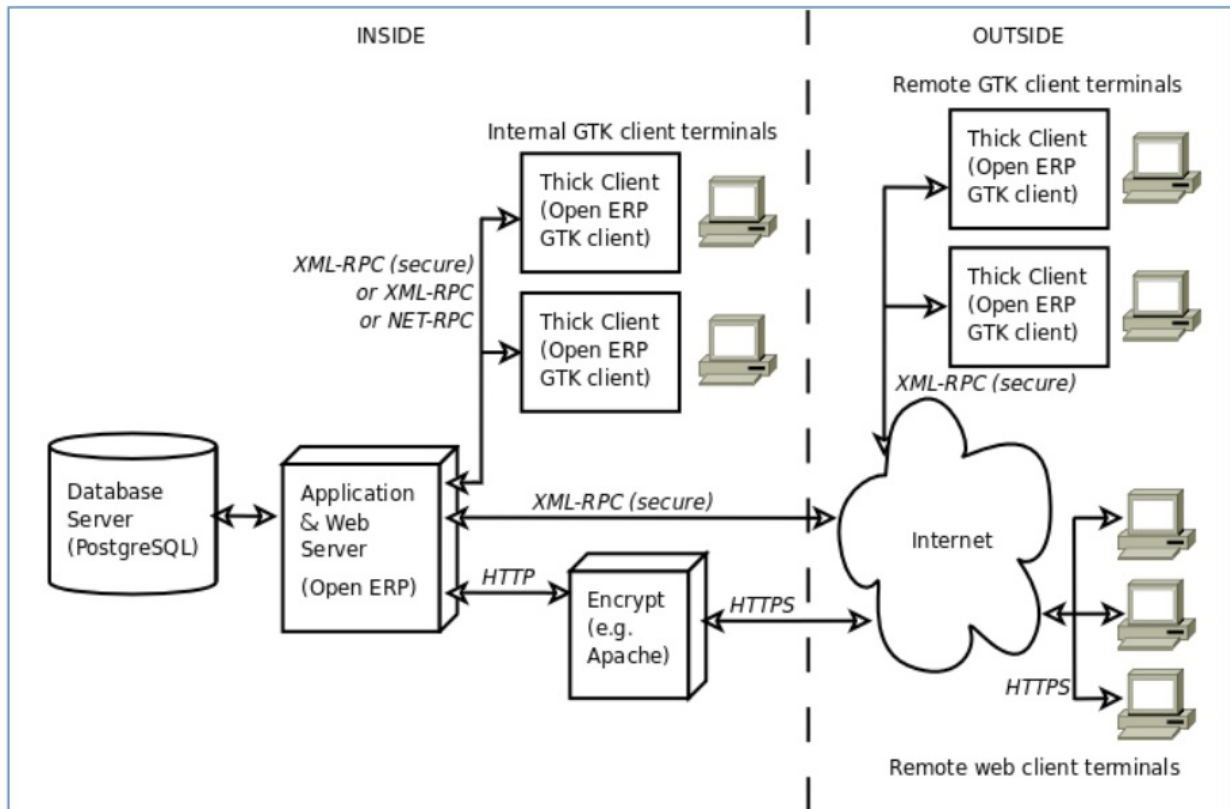
1. 激发一个 Action,Action 要求打开 account.invoice 对象。Action 中包含了对象、视图、域条件 (Domain, 如只显示未支付的凭证)等数据
2. 客户端询问服务端(通过 XML-RPC 协议)凭证对象定义了哪些视图,以及应显示什么数据。
3. 客户端呈现视图,展示数据。

# Achitectrue

---

## Odoo 架构

OpenERP 的安装部署,是典型的三层架构:数据库层、应用服务器层、客户层。



1. **数据库:**OpenERP 采用开源数据库 PostgreSQL。PostgreSQL 研发历史超过 30 年,最初是由著名的 IT 学院伯克利大学研发的。是世界上最好和最著名的大型开源数据库。OpenERP 利用了 PostgreSQL 的相当多的特性,两者绑定相当紧密。因此,OpenERP 目前只支持 PostgreSQL,不支持其他数据库。
2. **应用服务器:**OpenERP Server 承担应用服务器的角色。作为一个完整的企业应用服务器, OpenERP Server 包括数据访问层 ORM、界面展现层 View、Workflow 引擎、报表引擎,以及和其他系统集成的 Webservice 接口。



# Create the first Module

---

## 参考

<http://www.openerp-china.org/index.php?page=developer>

## 命令

```
./odoo.py scaffold module_name target(addons_path)
```

## 学习之前

对于OpenERP的开发，你只需要安装一个OpenERP（源码、ALL-IN-ONE），直接打开Python代码(推荐使用notepad++，editplus编辑器)，修改。

只需要有基本的Python语法知识还有就是Just Do It。

## 系统架构

OpenERP数据库使用的是PostgreSQL，有基于GTK的富客户端，也有基于网页的浏览器端

## 开发案例

通过此案例，你可以了解到：

1. 一个基本的OpenERP模块的构成
2. 字段的定义方法
3. 视图定义定义的方法（表单视图，列表视图，视图动作，菜单）

## 1. 写一个模块

### 需求

1. 输入和查询课程,把信息储存到课程对象里
2. 课程包含以下信息：名称，价格，天数，开始日期，教师，学员
3. 每个课程可以有多个学员，要记录学员的姓名、电话、电子邮件
4. 课程可以添加教材和作业等文档附件
5. 用户可以设置默认值以加速输入
6. 可以按名称查询课程，也可以用其他信息查找课程，并保存常用查询条件
7. 可以导出课程信息到excel文件，并支持导入
8. 可以按日期查看课程，并调整课程时间
9. 老师只能看到自己的课程
10. 模块名就叫做oecn\_training，然后它下面有四个文件，分别如下。

```
|--oecn_training
    |--__init__.py
    |--__openerp__.py
    |--lesson.py
    |--lesson_view.xml
```

## init.py

init.py文件是Python 的模块描述，因为OpenERP模块也是一个普通的Python模块。

```
# -*- coding: utf-8 -*-
import lesson #导入包含Python代码的所有文件和目录
```

## openerp.py

openerp.py文件（在6.0之前的版本也叫terp.py）它包含一个Python的字典声明这个模块的相关信息：模块名字，依赖关系，说明和组成。

```
# -*- coding: utf-8 -*-
# name: 模块名
#version: 模块版本
#description:模块说明
#author:作者
#website:网址
#depends:依赖的模块
#update_xml:模块更新的时候会读入的文件
#installable:可否安装
#category:模块类型
{
    "name" : "OECN Training",
    "version" : "1.0",
    "description" : 'OECN Training Demo',
    "author" : "Shine IT",
    "website" : "http://www.openerp.cn",
    "depends" : [],
    "update_xml" : ["lesson_view.xml"],
    "installable" : True,
    "category": 'Generic Modules/Others'
}
```

## lesson.py

```
# -*- coding: utf-8 -*-
from openerp.osv import fields, osv

class oecn_training_lesson(osv.osv):
    _name = 'oecn.training.lesson'
    _description = u'OECN 培训课程'
    _columns = {
        'name':fields.char( u'课程名',size=64,select=True),
        'date_start':fields.date(u'开始日期',select=True),
        'total_day':fields.float(u'总天数',digits=(16,1)),
```

```

        'teacher': fields.many2one('res.users', u'授课老师'),
        'students': fields.many2many('res.partner', string=u'学生'),
        'price': fields.float(u'价格', digits=(16, 2)),
    }

    oecn_training_lesson()

```

## lesson\_view.xml

```

<?xml version="1.0" encoding="utf-8"?>
<openerp>
    <data>
        <!-- 定义表单视图 -->
        <record model="ir.ui.view" id="oecn_training_lesson_form_view">
            <field name="name">课程表单</field>
            <field name="type">form</field>
            <field name="model">oecn.training.lesson</field>
            <field name="arch" type="xml">
                <form string="课程表单" version="7.0">
                    <group>
                        <field name="name"/>
                        <field name="date_start"/>
                        <field name="total_day"/>
                        <field name="price"/>
                        <field name="teacher"/>
                        <field name="students" colspan="4"/>
                    </group>
                </form>
            </field>
        </record>
        <!-- 定义列表视图 -->
        <record model="ir.ui.view" id="oecn_training_lesson_tree_view">
            <field name="name">课程列表</field>
            <field name="type">tree</field>
            <field name="model">oecn.training.lesson</field>
            <field name="arch" type="xml">
                <tree string="课程列表" version="7.0">
                    <field name="name"/>
                    <field name="date_start"/>
                    <field name="teacher"/>
                    <field name="price" sum="合计"/>
                </tree>
            </field>
        </record>
        <!-- 定义视图动作 -->
        <record model="ir.actions.act_window" id="action_oecn_training_lesson">
            <field name="name">课程</field>
            <field name="res_model">oecn.training.lesson</field>
            <field name="view_type">form</field>
            <field name="view_mode">form,tree</field>
            <field name="view_id" ref="oecn_training_lesson_tree_view"/>
        </record>
        <!-- 定义菜单 -->
        <menuitem id="oecn_menu" name="OECN"/>
        <menuitem id="oecn_training_menu" name="OECN Training" parent="oecn_menu"/>
        <menuitem id="oecn_training_lesson_menu" name="OECN Training Lesson" parent="oecn_training_menu"/>
    </data>

```

```
</openerp>
```

把oecn\_training这个目录复制到openerp的addons目录下

登录OpenERP

1. 确保在扩展视图下（右上角小齿轮->首选项->扩展）

!!7.0 版本OpenERP 可以忽略此步直接 更新模块列表，如果没有 更新模块列表 菜单，请用 admin 用户登陆修改对应登陆用户的权限，勾选"技术特性".

2. 更新模块列表（设置->模块->更新模块列表）
3. 搜索自己的模块（设置->模块->模块）
4. 安装后重启服务器

## 2. 通过继承修改一个模块

需求

我们想添加一个教室对象，而且给每个课程都分配一个教室。

分析

有两个办法可以做到：一、修改上面的程序。虽然上面是新的模块可能我们直接修改没有特别的问题，但是如果是系统默认模块怎么办呢，如果我直接修改了源码，那么以后模块的升级会造成各种不可预料的问题,而且同一个源码会影响到所有的的帐套。所以我们建议通过继承来修改模块。我们会以一个oecn\_training\_classroom来继承oecn\_training

```
|--oecn_training_classroom
    |--__init__.py
    |--__openerp__.py
    |--lesson.py #继承oecn.training.lesson对象的文件
    |--classroom.py
    |--lesson_view.xml
    |--classroom_view.xml
```

### init.py

```
# -*- coding: utf-8 -*-
#导入包含Python代码的所有文件和目录
import lesson
import classroom
```

### openerp.py

```
# -*- coding: utf-8 -*-
# -*- coding: utf-8 -*-
# name: 模块名
#version: 模块版本
#description:模块说明
#author:作者
#website:网址
#depends:#依赖的模块,此模块我们会继承oecn_training所以必须依赖他
#update_xml:模块更新的时候会读入的文件 #需要继承的视图
#installable:可否安装
#category:模块类型
{
    "name" : "OECN Training Classroom",
    "version" : "1.0",
    "description" : "OECN Training Demo -- ClassRoom",
    "author" : "Shine IT",
    "website" : "http://www.openerp.cn",
    "depends" : ['oecn_training'],
    "update_xml" : [
        "lesson_view.xml" ,
        "classroom_view.xml",],
    "installable" : True,
    "category": "Generic Modules/Others"
}
```

## lesson.py

```
# -*- coding: utf-8 -*-
from openerp.osv import fields, osv

class oecn_training_lesson(osv.osv):
    _inherit = 'oecn.training.lesson' #要继承的对象的_name
    _columns = {
        'classroom_id':fields.many2one('oecn.training.classroom','教室'),#添加一个教室属性, 为
    }
    oecn_training_lesson()
```

## classroom.py

```
# -*- coding: utf-8 -*-
from openerp.osv import fields, osv

class oecn_training_classroom(osv.osv):
    _name = 'oecn.training.classroom'
    _description = u'OECN 教室'
    _columns = {
        'number':fields.char(u'编号', size=64, select=True),
        'capacity':fields.integer(u'容纳人数', select=True),
        'location':fields.char(u'地点', size=125, select=True),
    }
    oecn_training_classroom()
```

## lesson\_view.xml

```
<?xml version="1.0" encoding="utf-8"?>
<openerp>
  <data>
    <!--OECD 教室-->
      <record model="ir.ui.view" id="oecn_training_lesson_from_inherit_classroom_vi
        <field name="name">课程教室继承视图</field>
        <field name="type">form</field>
        <field name="model">oecn.training.lesson</field>
        <field name="inherit_id" ref="oecn_training.oecn_training_lesson_form_view"/
        <field name="arch" type="xml">
          <field name="name" position="after">
            <field name="classroom_id"/>
          </field>
        </field>
      </record>
    </data>
  </openerp>
```

上面就是继承视图的例子。

**id**: XML ID每个模块里必须是唯一的，如果存在多个，则会后者覆盖前者。

**type**:需要继承的视图的类型。

**model**:需要继承的对象的名字

**inherit\_id**:需要继承的视图的ID，格式为：模块名.XML ID

```
<field name="name" position="after">
<field name="classroom_id"/>
</field>
```

意思是在要继承的视图的name字段后（position="after"），添加一个教室字段。

## classroom\_view.xml

```
<?xml version="1.0" encoding="utf-8"?>
<openerp>
  <data>
    <!--OECD 教室-->
      <record model="ir.ui.view" id="oecn_training_classroom_from_view">
        <field name="name">教室</field>
        <field name="type">form</field>
        <field name="model">oecn.training.classroom</field>
        <field name="arch" type="xml">
          <field name="number"/>
          <field name="capacity"/>
          <field name="location" />
        </field>
      </record>
    </data>
  </openerp>
```

```

        </field>
    </record>
    <!-- 定义列表视图 -->
    <record model="ir.ui.view" id="oecn_training_classroom_tree_view">
        <field name="name">教室列表</field>
        <field name="type">tree</field>
        <field name="model">oecn.training.classroom</field>
        <field name="arch" type="xml">
            <field name="number"/>
            <field name="capacity"/>
            <field name="location" />
        </field>
    </record>
    <!-- 定义视图动作 -->
    <record model="ir.actions.act_window" id="action_oecn_training_classroom">
        <field name="name">教室</field>
        <field name="res_model">oecn.training.classroom</field>
        <field name="view_type">form</field>
        <field name="view_mode">form,tree</field>
        <field name="view_id" ref="oecn_training_classroom_tree_view"/>
    </record>
    <menuitem id="oecn_training_classroom_menu" name="OECN Training Classroom" parent
    </data>
</openerp>

```

## Model

---

<http://stackoverflow.com/questions/14608775/how-to-filter-datas-in-openerp-using-domain-list/14618598#14618598>

Your domain syntax is wrong.

It should be `[('user_ids', '=', user.id)]`

Each tuple in the search domain needs to have 3 elements, in the form: ('field\_name', 'operator', value), where:

field\_name must be a valid name of field of the object model, possibly following many-to-one relationships using dot-notation, e.g 'street' or 'partner\_id.country' are valid values.

operator must be a string with a valid comparison operator from this list: =, !=, >, >=, <, <=, like, ilike, in, not in, child\_of, parent\_left, parent\_right. The semantics of most of these operators are obvious. The child\_of operator will look for records who are children or grand-children of a given record, according to the semantics of this model (i.e following the relationship field named by self.\_parent\_name, by default parent\_id).

value must be a valid value to compare with the values of field\_name, depending on its type.

Domain criteria can be combined using 3 logical operators than can be added between tuples: '&' (logical AND, default), '|' (logical OR), '!' (logical NOT). These are prefix operators and the arity of the '&' and '|' operator is 2, while the arity of the '!' is just 1. Be very careful about this when you combine them the first time.

Here is an example of searching for Partners named ABC from Belgium and Germany whose language is not english ::

```
[('name','=', 'ABC'), '!', ('language.code', '=', 'en_US'), '|', ('country_id.code', '=', 'be'), ('country_id.code', '=', 'de')]
```

The '&' is omitted as it is the default, and of course we could have used '!=' for the language, but what this domain really represents is::

```
(name is 'ABC' AND (language is NOT english) AND (country is Belgium OR Germany))
```



## Odoo 对象

### 一切都是对象

OpenERP 的所有资源(Resource)都是对象,如 menus, actions, reports, invoices, partners 等等。换言之,在 OpenERP 中,一个菜单项,一个弹出窗口,其实都是一条数据库记录。OpenERP 运行时,从数据库读出“菜单项”记录,根据该记录的信息,在屏幕上显示菜单项及其子菜单项。因此,理论上,可以不写代码,而是直接修改 OpenERP 的数据库而编写功能菜单、查询窗口、动作按钮等实现业务功能开发。实际开发中,通常是编写 XML 文件,导入菜单、窗口、动作等编程元素,实现功能开发。XML 文件比直接修改数据库或编写 SQL 语句更容易使用一些。

OpenERP 通过自身实现的对象关系映射(ORM,object relational mapping of a database)访问数据库。OpenERP 的对象名是层次结构的,就是说可以使用"."访问树状对象,如: ·account.invoice : 表示财务凭证对象。 ·account.invoice.line : 表示财务凭证对象中的一个明细行对象。

通常,对象名中,第一级是模块名,如: account, stock, sale 等。比之直接用 SQL 访问数据库,OpenERP 的对象的优势有,

1. 直接使用对象的方法增、删、改数据库记录。因为 OpenERP 在基类对象中实现了常规的增、删、改方法,因而,普通对象中不需要写任何方法和代码就具备增、删、改数据库记录的功能。
2. 对于复杂对象,只需操作一个对象即可访问多张数据表。如 partner 对象,它的信息实际上存储在多张数据表中(partner address, categories, events 等等),但只要通过"."操作即可访问所有关联表(如,partner.address.city),简化了数据库访问。

注意,在其他编程语言或开发平台(如 Java or JavaEE)中,一个对象(Object)通常和数据库中一条记录(Record)相对应。但是,OpenERP 的对象其实是一个 Class,它和一个数据表(Table)对应,而不是和一个记录(Record)对应。在 OpenERP 中,数据库记录(Record)通常叫资源(Resource)。因为 Object 操作的是数据表,OpenERP 的对象的方法(Method)中,几乎每个方法都带有参数 ids,该参数是资源(Resource or Record)的 ID(在 OpenERP 中 ID 是主键)列表,通过该 ids 就可以操作具体的 Record 了。

### 访问 OpenERP 对象

OpenERP 提供了三种方式执行对象的方法(Method),每种方式都是先取得对象,然后调用对象的方法。三种方式是,

1. 直接使用对象
2. 通过 **netservice** 使用对象
3. 通过 **xmlrpc** 使用对象。

#### 直接使用对象

这种方式最简单,这种方式只能在 OpenERP Server 端使用,编写 OpenERP 的模块时候,通常使用这种方式。这个方式的内部实现原理是,OpenERP 加载模块(不是安装,是启动时加载已安装模块)时,会将创建模块中的对象实例,对象实例以对象名为关键字,存储在对象池(pool)中。此方式是从 pool 中取得对象,而后

调用对象的方法。

这个方式的一般调用形式是：

```
obj=self.pool.get('name_of_the_object')
obj.name_of_the_method(parameters_for_that_method)
```

第一行代码从对象池中取得对象实例,第二行代码调用对象的方法。

## Netservice 方式

这个方式和直接使用对象的方式是类似的,只是不以对象的方式呈现,而是以“服务”(Service)的方式呈现。这种方式也只能在 OpenERP Server 端使用,即调用程序和 OpenERP Server 程序在同一个 Python 虚拟机上运行。这个方式的内部实现原理是,类似对象池,OpenERP 有一个全局变量的服务池: SERVICES,该变量位于 bin\netsvc.py。有一些对象,它在创建时( \_\_init\_\_ 方法中)将自己提供的服务登记在服务池中,并暴露自己的服务方法(即该服务可供调用的method)。和对象池不同的是,服务可以有选择性的暴露自己的方法。OpenERP 的工作流(Workflow)、报表(Report)都以服务的形式暴露自己的方法,关于 OpenERP 可供使用的服务有哪些,将在以后介绍。这个方式调用形式如下:

```
service = netsvc.LocalService("object_proxy")
result = service.execute(user_id, object_name, method_name, parameters)
```

第一行指定服务名取得服务,"object\_proxy"是 osv.osv 对象初始化时注册的一个服务,这个服务可用于调用 OpenERP 的几乎所有对象(准确的说是所有从 osv.osv 派生的对象)。“object\_proxy”服务用于调用对象的方法。第二行是其调用格式,execute 是该服务暴露的一个服务方法,该方法的参数说明如下:

- user\_id: 用户 id,以用户名、密码登录后取得的 id。
- object\_name: 对象名,欲访问的对象的名称,如"res.partner"等。
- method\_name: 方法名,欲调用的方法的名称,如"create"等。
- parameters: 方法的参数。

## XML-RPC 方式

这个方式相当灵活,它以 HTTP 协议远程访问对象,因此,能在本机、局域网、广域网范围调用 OpenERP 的对象的方法。该方式的调用形式是:

```
sock = xmlrpclib.ServerProxy('http://server_address:port_number/xmlrpc/object')
result = sock.execute(user_id, password, object_name, method_name, parameters)
```

参数说明如下:

- server\_address: 运行 OpenERP Server 的机器的 IP 或域名。
- port\_number: OpenERP Server 的 xmlrpc 调用端口,缺省情况是 8069。
- execute 的参数和 Netservice 方式相同,只是多了个 password 参数,该参数即用户的登录密码。

XML-RPC 方式参考例子。这个例子以 xmlrpc 方式调用 OpenERP 的对象 res.partner,在数据库 中插入一条业务伙伴及其联系地址记录。因为含有中文,测试时注意代码文件保存成 utf-8 格式:

```
import xmlrpclib #导入 xmlrpc 库,这个库是 python 的标准库。
username = 'admin' #用户登录名
pwd = '123' #用户的登录密码,测试时请换成自己的密码
dbname = 'case1' #数据库帐套名,测试时请换成自己的帐套名
# 第一步,取得 uid
sock_common = xmlrpclib.ServerProxy ('http://localhost:8069/xmlrpc/common')
uid = sock_common.login(dbname, username, pwd)
#replace localhost with the address of the server
sock = xmlrpclib.ServerProxy('http://localhost:8069/xmlrpc/object')
# 调用 res.partner 对象的 create 方法在数据库中插入一个业务伙伴
partner = {
    'name': 'shine-it',
    'lang': 'zh_CN',
}
partner_id = sock.execute(dbname, uid, pwd, 'res.partner', 'create', partner)
开源智造咨询有限公司(OSCG) - OpenERP 7.0 开发教程 参考: Document Reference 版本 Draft
页 21/56
# 下面再创建业务伙伴的联系地址记录
address = {
    'partner_id': partner_id,
    'type': 'default',
    'street': '浦东大道 400 号',
    'zip': '200000',
    'city': '上海市',
    'phone': '021-88888888',
}

address_id = sock.execute(dbname, uid, pwd, 'res.partner.address', 'create', address)
```

## 再议 OPENERP 的对象

通过上面的解说,可以这样通俗的理解 OpenERP 的对象:

每个对象就是一个代码块,包含了数据表操作(增删改查)的代码。OpenERP Server 好比是一个代码池,里面装满了代码块。

通过对象池、服务池、xmlrpc 等方式,可以取得代码块位置(或者专业一点,叫指针),然后调用代码块的方法,操作数据库。对象的代码什么时候装入“代码池”呢?每个对象定义的后面都有一行:name\_of\_the\_object(),这一行实际上是创建对象实例,实例创建好以后就装到了代码池,这个装入的过程在对象的基类(osv.osv)中完成。

对象的基类(osv.osv)已实现了增删改查等常规数据表操作方法,因此,只要定义好对象的字段,即使不写任何代码,该对象已经具备增删改查数据表的能力。

## 对象定义的属性

---

Odoo 的对象定义的一般形式如下:

```
class name_of_the_object(osv.osv):
    _name = 'xxx'
    .....
    name_of_the_object()
    #Sample:
    class qingjd(osv.osv):
        _name = 'qingjia.qingjd'
        _description = '请假单'
        _columns = {
            'shenqr': fields.many2one('hr.employee', '申请人', required=True),
        }

    qingjd()
```

对象定义的完整属性如下:

必须属性:

- `_name`
- `_columns *`

可选属性:

- `_table`
- `_description`
- `_defaults`
- `_order`
- `_rec_name`
- `_auto`
- `_constraints`
- `_sql_constraints`
- `_inherit`
- `_inherits`

## 必须属性

---

必须属性:

- `_name`
- `_columns` \*

## 可选属性

可选属性:

- `_table`
- `_description`
- `_defaults`
- `_order`
- `_rec_name`
- `_auto`
- `_constraints`
- `_sql_constraints`
- `_inherit`
- `_inherits`

下面详细解说各个属性。

### `_auto`

是否自动创建对象对应的 Table,缺省值为: `True`。当安装或升级模块时,OpenERP 会自动在 数据库中为模块中定义的每个对象创建相应的 Table。当这个属性设为 `False` 时,OpenERP不会自动创建 Table,这通常表示数据库表已经存在。

例如,当对象是从数据库视图(View)中读取数据时,通常设为 `False`。当 `_auto` 的值为 `"False"`时,OE 不会自动在数



### `_columns`

定义对象的字段,系统会为这里定义的每个字段在数据库表中创建相应的字段。关于字段(Fields)的定义,参见后文。

### `_constraints`

定义于对象上的约束(constraints),通常是定义一个检查函数,关于约束的详细说明,参见后文。

`_constraints` 可以灵活定义 OpenERP 对象的约束条件,当创建或更新记录时,会触发该条件,如果条件不符合,则弹出错误信息,拒绝修改。

`_constraints` 的定义格式:

```
[(method, 'error message', list_of_field_names), ...]
```

- `method`: 是对象的方法,该方法的格式为:

```
def _name_of_the_method(self, cr, uid, ids): -> True|False
```

- error message: 不符合检查条件(method 返回 False)时的错误信息。
- list\_of\_field\_names: 字段名列表,这些字段的值会出现在 error message 中。通常列出能帮助用户理解错误的字段。

`_constraints` 的例子:

```
def _constraint_sum(self, cr, uid, ids):
    cr.execute('SELECT a.currency_id
                FROM account_move m, account_move_line l, account_account a
                WHERE m.id=l.move_id AND l.account_id=a.id AND m.id IN ('+', '.join(map(str, ids))
                GROUP BY a.currency_id')

    if len(cr.fetchall()) >= 2:
        return True

    cr.execute('SELECT abs(SUM(l.amount))
                FROM account_move m LEFT JOIN account_move_line l ON (m.id=l.move_id)
                WHERE m.id IN ('+', '.join(map(str, ids))+')')

    res = cr.fetchone()[0]
    return res < 0.01

_constraints = [
    (_constraint_sum, 'Error: the sum of all amounts should be zero.', ['name'])
]
```

## `_defaults`

定义字段的缺省值。当创建一条新记录(record or resource)时,记录中各字段的缺省值 在此定义。

`_defaults` 属性用于定义字段的缺省值,其格式为:

```
_defaults:
{
    'name_of_the_field':function, ...
}
```

function 的返回值作为'name\_of\_the\_field'字段的缺省值。function 格式是:function(obj, cr, uid, context),返回值必须是简单类型,如 boolean, integer, string 等。下面是 `_defaults` 的例子。

```
_defaults = {
    'date_order': lambda *a: time.strftime('%Y-%m-%d'),
    'state': lambda *a: 'draft',
    'user_id': lambda obj, cr, uid, context: uid
}
```

lambda 是 Python 的行函数,"lambda obj, cr, uid, context: uid"等同于下述函数: def func(obj, cr, uid, context): return uid

在 V6 中字典的值可以不是函数,就比如在 V5 中我们必须这样来定义: `_defaults= { 'state' : lambda *a: 'draft'}`

可选属性

而在 V6 中可以这样来: `_defaults = {'state': 'draft'}`

另外一个例子：设置默认图片

file: object.py

```
def _get_default_image(self, cr, uid, context=None):
    image_path = get_module_resource('it', 'static/description/', 'default_image_equi
    return tools.image_resize_image_big(open(image_path, 'rb').read()).encode('base64'

_defaults = {
    'pin': _get_pin,
    'image': _get_default_image,
}
```

## **`_description`**

对象说明性文字,任意文字。

## **`_log_access`**

是否自动在对应的数据表中增加 `create_uid`, `create_date`, `write_uid`, `write_date` 四个字段,缺省值为 `True`, 即字段增加。这四个字段分布记录 `record` 的创建人,创建日期,修改人,修改日期。这四个字段值可以用对象的方法(`perm_read`)读取。

## **`_name`**

对象的唯一标识符,必须是全局唯一。这个标识符用于存取对象,其格式通常是 `"ModuleName.ClassName"`, 对应的,系统会字段创建数据库表 `"ModuleName_ClassName"`。

## **`_order`**

定义 `search()` 和 `read()` 方法的结果记录的排序规则,和 SQL 语句中的 `order` 类似,缺省值是 `id`,即按 `id` 升序排序。详细说明参见后文。

`_order` 在对象的 `search` 或 `read` 方法中的 `select` 语句上加上 `"Order"` 子句,如 `_order = 'name desc, account_id'`,对应 SQL 文的 `Order` 子句: `order by name desc, account_id`。

## **`_rec_name`**

参考:

- <http://dirtyhandsphp.blogspot.jp/2014/09/odooopenerp-recname-and-nameget.html>
- <https://www.odoo.com/forum/help-1/question/rec-name-predefined-fields-17172>

标识 `record name` 的字段。缺省情况(`name_get` 没被重载的话)方法 `name_get()` 返回本字段值。

`_rec_name` 通常用于记录的显示,例如,销售订单中包含业务伙伴,当在销售订单上显示业务伙伴时,系统缺省的是显示业务伙伴记录的 `_rec_name`。



## `_sequence`

宋教授讲解

数据库表的id字段的序列采集器,缺省值为: None。OpenERP 创建数据库表时,会自动增加 id 字段作为主键,并自动为该表创建一个序列(名字通常是“表名\_id\_seq”)作为 id 字段值的采集器。如果想使用数据库中已有的序列器,则在此处定义序列器名。

## `_sql_constraints`

定义于对象上的约束(constraints),和 SQL 文中的约束类似。

`_sql_constraints` 定义数据表的约束条件,其格式如下例所示

```
_sql_constraints = [
    ('code_company_uniq', 'unique (code,company_id)', 'The code of the account must be unique
    ]
```

本例的 `_sql_constraints` 会在数据表中增加下述约束:

```
CONSTRAINT ObjectName_code_company_uniq UNIQUE(code, company_id)
```

## `_table`

待创建的数据库表名,缺省值是和 *name* 一样,只是将 "." 替换成 ""。

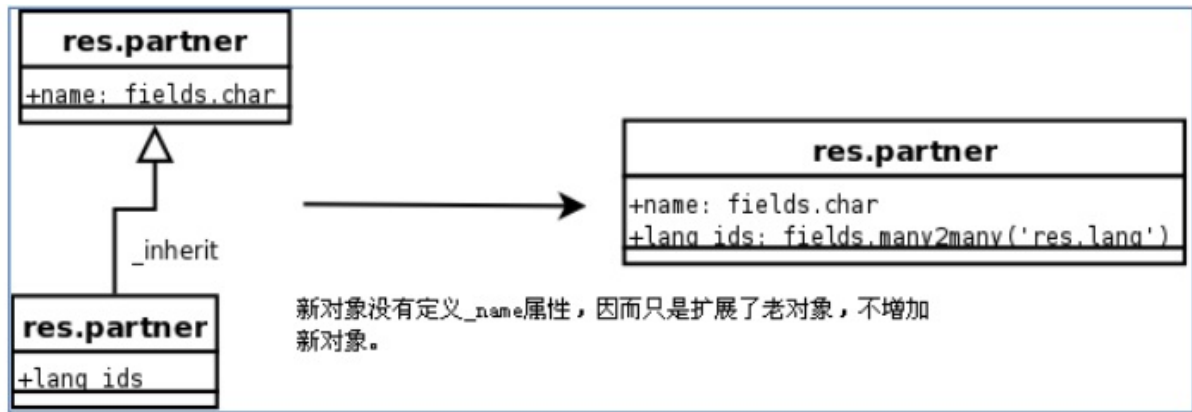
## `_inherits`

### `_inherit`

`_inherits` 和 `_inherit` 都用于对象的继承,详细说明参见后文。

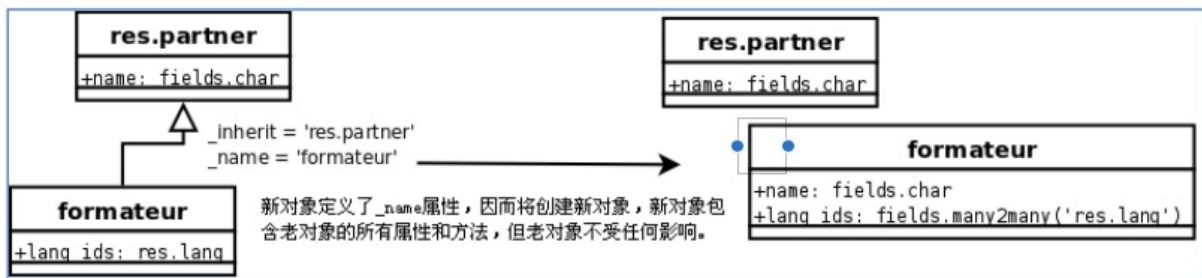
`_inherit` 继承有两种情况:

1. 如果子类中不定义 `_name` 属性,则相当于在父类中增加一些字段和方法,并不创建新对象。
2. 如果子类中定义 `_name` 属性,则创建一个新对象,新对象拥有老对象的所有字段和方法,老对象不受任何影响。两种情况的示例及继承关系的图示见下面。



```
class res_partner_add_langs(osv.osv):
    _inherit = 'res.partner'
    _columns = {
        'lang_ids' : fields.many2many('res.lang', 'res_lang_partner_rel', 'partner_id', 'lang_id')
    }

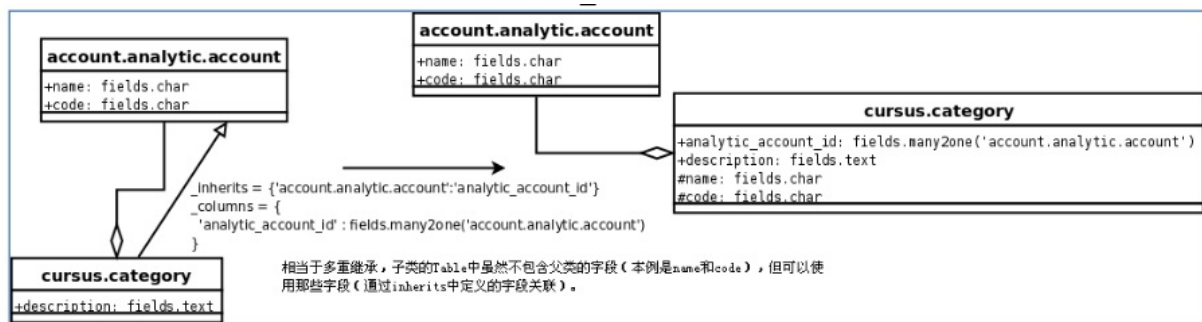
res_partner_add_langs()
```



```
class formateur(osv.osv):
    _name = 'formateur'
    _inherit = 'res.partner'
    _columns = {
        'lang_ids' : fields.many2many('res.lang', 'res_lang_partner_rel', 'partner_id', 'lang')
    }

formateur()
```

inherits 相当于多重继承。子类通过\_inherits 中定义的字段和各个父类关联,子类不拥有父类的 字段,但可以直接操作父类的所有字段和方法。\_inherits 的示例及图示见下图。



```

class curso_category(osv.osv):
    _name = 'curso.category'
    _inherits = {'account.analytic.caccount': 'analytic_caccount_id'}
    _columns = {
        'analytic_caccount_id' : fields.many2one('account.analytic.caccount', 'ID'), }

curso_category()

```

例子2:

参考： [https://doc.odoo.com/6.0/developer/2\\_5\\_Objects\\_Fields\\_Methods/object\\_inherits/](https://doc.odoo.com/6.0/developer/2_5_Objects_Fields_Methods/object_inherits/)

```

class tiny_object(osv.osv)
    _name = 'tiny.object'
    _table = 'tiny_object'
    _inherits = {
        'tiny.object_a': 'object_a_id',
        'tiny.object_b': 'object_b_id',
        ...,
        'tiny.object_n': 'object_n_id'
    }
    (...)

```

参考： Whats the difference between inherit and inherits? <https://www.odoo.com/forum/how-to-developers-13/whats-the-difference-between-inherit-and-inherits-52205>

[https://www.odoo.com/files/memento/OpenERP\\_Technical\\_Memento\\_latest.pdf](https://www.odoo.com/files/memento/OpenERP_Technical_Memento_latest.pdf)

## OPENERP 对象字段的定义

OpenERP 对象支持的字段类型有,基础类型:char, text, boolean, integer, float, date, time, datetime, binary; 复杂类型:selection, function, related;关系类型:one2one, one2many, many2one, many2many。下面逐一说明。

- boolean: 布尔型(true, false)
- integer: 整数。
- float: 浮点型,如 'rate': fields.float('Relative Change rate',digits=(12,6)), digits 定义整数部分和小数部分的位数。
- char: 字符型,size 属性定义字符串长度。
- text: 文本型,没有长度限制。
- date: 日期型
- datetime: 日期时间型
- binary: 二进制型
- function: 函数型,该类型的字段,字段值由函数计算而得,不存储在数据表中。其定义格式为:

```
fields.function(fnct, arg=None, fnct_inv=None, fnct_inv_arg=None, type='float', fnct_search
```

type 是函数返回值的类型。

method 为 True 表示本字段的函数是对象的一个方法,为 False 表示是全局函数,不是对象的方法。如果 method=True,obj 指定 method 的对象。

fnct 是函数或方法,用于计算字段值。

```
#如果 method = true, 表示 fnct 是对象的方法,其格式如下:
def fnct(self, cr, uid, ids, field_name, args, context)
#否则,其格式如下:
def fnct(cr, table, ids, field_name, args, context)
#ids 是系统传进来的当前存取的 record id。field_name是本字段名,当一个函数用于多个函数字段类型时,本参数
```

fnct\_inv 是用于写本字段的函数或方法。

```
#如果 method = true, 其格式是:
def fnct_inv(self, cr, uid,ids, field_name, field_value, args, context)
#否则格式为:
def fnct_inv(cr, table, ids, field_name, field_value, args, context)
```

fnct\_search 定义该字段的搜索行为。

```
# 如果 method = true, 其格式为:
def fnct_search(self, cr, uid,obj, field_name, args)
```

```
# 否则格式为:
def fcmt_search(cr, uid, obj, field_name, args)
```

store 表示是否希望在数据库中存储本字段值,缺省值为 False。不过 store 还有一个增强形式,格式为 store={ 'object\_name':(function\_name,['field\_name1','field\_name2'],priority)} ,其含义是,如果对象'object\_name'的字段['field\_name1','field\_name2']发生任何改变,系统将调用函数 function\_name,函数的返回结果将作为参数(arg)传送给本字段的主函数,即 fcmt。

- selection: 下拉框字段。定义一个下拉框,允许用户选择值。如:'state':  
fields.selection(((('n','Unconfirmed'),('c','Confirmed'))),'State', required=True),这表示 state 字段有两个选项('n','Unconfirmed')和('c','Confirmed')。
- one2one: 一对一关系,格式为:fields.one2one(关联对象 Name, 字段显示名, ... )。在 V5.0 以后的版本中不建议使用,而是用 many2one 替代。
- many2one: 多对一关系,格式为:fields.many2one(关联对象 Name, 字段显示名, ... )。可选参数有: ondelete,可选值为"cascade"和"null",缺省值为"null",表示 one 端的 record 被删除后,many 端的 record 是否级联删除。
- one2many: 一对多关系,格式为:fields.one2many(关联对象 Name, 关联字段, 字段显示名, ... ),例:  
'address': fields.one2many('res.partner.address', 'partner\_id', 'Contacts')。
- many2many: 多对多关系。例如:

```
'category_id':fields.many2many('res.partner.category','res_partner_category_rel','partner_id','category_id','Categories')
```

表示以多对多关系关联到对象 res.partner.category,关联表为'res\_partner\_category\_rel',关联字段为 'partner\_id'和'category\_id'。当定义上述字段时,OpenERP 会自动创建关联表为 'res\_partner\_category\_rel',它含有关联字段'partner\_id'和'category\_id'。

- reference: 引用型,格式为:fields.reference(字段名, selection, size, ... )。其中 selection 是:
  1. 返回 tuple 列表的函数
  2. 表征该字段引用哪个对象(or model)的 tuples 列表。reference 字段在数据库表中的存储形式是(对象名,ID),如(product.product,
  3. 表示引用对象 product.product(数据表 product\_product)中 id=3 的数据。reference 的例子:

```
def _links_get(self, cr, uid):
    cr.execute('select object,name from res_request_link order by priority')
    return cr.fetchall()
...
'ref':fields.reference('Document Ref 2', selection=_links_get, size=128), ...
```

上例表示,字段 ref 可以引用哪些对象类型的 resource,可引用的对象类型从下拉框选择。下拉框的选项由函数\_links\_get 返回,是 (object,name)对的列表,如[("product.product","Product"), ("account.invoice","Invoice"), ("stock.production.lot","Production Lot")]

- **related:** 关联字段,表示本字段引用关联表中的某字段。格式为:fields.related(关系字段,引用字段,type, relation, string, ...),关系字段是本对象的某字段(通常是 one2many or many2many),引用字段是通过关系字段关联的数据表的字段,type 是引用字段的类型,如果 type 是 many2one or many2many, relation 指明关联表。例子如下:

```
'address': fields.one2many('res.partner.address', 'partner_id', 'Contacts'), 'city': f
string='Country'),
```

这里,city 引用 address 的 city 字段,country 引用 address 的 country 对象。在 address 的关联对象 res.partner.address 中,country\_id 是 many2one 类型的字段,所以 type='many2one', relation='res.country'。

- **property:** 属性字段,下面以具体例子解说 property 字段类型。 'propertyproduct\_pricelist':  
fields.property('product.pricelist', type='many2one', relation='product.pricelist',string="Sale Pricelist",  
method=True, view\_load=True, group\_name="Pricelists Properties") 这个例子表示,本对象通过字段 'property\_product\_pricelist'多对一(type='many2one')关联到对象 product.pricelist(relation='product.pricelist')。和 many2one 字段类型不同的是,many2one 字段会在本对象中创建数据表字段'property\_product\_pricelist',property 字段类型不会创建数据表字段 'property\_product\_pricelist'。property 字段类型会从数据表 ir.property 中查找 name='property\_product\_pricelist'(即字段定义中的'product.pricelist'加上前缀 property,并将"."替换成""作为 name)且 company\_id 和本对象相同的记录,从该记录的 value 字段(value 字段类型为 reference)查得关联记录,如(product.pricelist,1),表示本对象的 resource 多对一关联到对象 product.pricelist 的 id=1 的记录。也就是说,property 字段类型通过 ir.property 间接多对一关联到别的对象。property 字段类型基本上和 many2one 字段类型相同,但是有两种情况优于 many2one 字段。其一是,例如,当有多条记录通过 ir.property 的 name='property\_product\_pricelist'的记录关联到记录 (product.pricelist,1),此时,如果希望将所关联关系都改成关联到记录(product.pricelist,2)。如果是 many2one 类型,不写代码,很难完成此任务,是 property 字段的话,只要将 ir.property 中的 value 值 (product.pricelist,1)改成(product.pricelist,2),则所关联关系都变了。修改 ir.property 的 value 值 可以在系统管理下的菜单 Configuration --> Properties 中修改。其二是,例如,同一业务伙伴,但希望 A 公司的用户进来看到的该业务伙伴价格表为 pricelistA,B 公司的用户进来看到的该业务伙伴价格表为 pricelistB,则 many2one 类型达不到该效果。property 类型通过 ir.property 中的记录关联时加上了 company\_id 的条件,因此可以使得不同公司的员工进来看到不同的关联记录。由于 property 类型通过 ir.property 关联,因此,每个 property 类型的字段都必须在 ir.property 中有一条关联记录。这可以在安装时导入该条记录,参考代码如下: property\_product\_pricelist

## 字段定义的参数

字段定义中可用的参数有, `change_default`, `readonly`, `required`, `states`, `string`, `translate`, `size`, `priority`, `domain`, `invisible`, `context`, `selection`。

- `change_default`: 别的字段的缺省值是否可依赖于本字段,缺省值为:False。 例子(参见 `res.partner.address`),

```
'zip': fields.char('Zip', change_default=True, size=24),
```

这个例子中,可以根据 `zip` 的值设定其它字段的缺省值,例如,可以通过程序代码,如果 `zip` 为 200000 则 `city` 设为“上海”,如果 `zip` 为 100000 则 `city` 为“北京”。

参考: <http://cn.openerp.cn/openerp-7-set-dafault/>

**readonly**: 本字段是否只读,缺省值:False。

**required**: 本字段是否必须的,缺省值:False。

??? **states**: 定义特定 state 才生效的属性,格式为: `{'name_of_the_state': list_of_attributes}`,其中 `list_of_attributes` 是形如 `[('name_of_attribute', value), ...]` 的 tuples 列表。例子(参见 `account.transfer`):

```
'partner_id': fields.many2one('res.partner', 'Partner', states={'posted': [('readonly', True)]})
```

**string**: 字段显示名,任意字符串。

**translate**: 本字段值(不是字段的显示名)是否可翻译,缺省值:False。

**size**: 字段长度。

**priority**:

**domain**: 域条件,缺省值:[]。在 `many2many` 和 `many2one` 类型中,字段值是关联表的 id,域条件 用于过滤关联表的 record。例子:

```
'default_credit_account_id': fields.many2one('account.account', 'Default Credit Account',
```

本例表示,本字段关联到对象 (`'account.account'`) 中的 type 不是 `'view'` 的 record。

**invisible**: 本字段是否可见,即是否在界面上显示本字段,缺省值 True。

**selection**: 只用于 reference 字段类型,参见前文 reference 的说明。

## OpenERP 对象预定义方法

---

每个 OpenERP 的对象都有一些预定义方法,这些方法定义在基类 `osv.osv` 中。这些预定义方法有:

### 基本方法

---

- `create`,
- `search`,
- `read`,
- `browse`,
- `write`,
- `unlink`。

```
def create(self, cr, uid, vals, context={})
```

```
def search(self, cr, uid, args, offset=0, limit=2000)
```

```
def read(self, cr, uid, ids, fields=None, context={})
```

```
def browse(self, cr, uid, select, offset=0, limit=2000)
```

```
def write(self, cr, uid, ids, vals, context={})
```

```
def unlink(self, cr, uid, ids)
```

### 缺省值存取方法

---

- `default_get`,
- `default_set`。

```
def default_get(self, cr, uid, fields, form=None, reference=None) def
```

```
default_set(self, cr, uid, field, value, for_user=False)
```

### 特殊字段操作方法

---

- `perm_read`
- `perm_write`

```
def perm_read(self, cr, uid, ids) def perm_write(self, cr, uid, ids, fields)
```

### 字段(fields)和视图(views)操作方法

---

- `fields_get`
- `distinct_field_get`
- `fields_view_get`



```
def fields_get(self, cr, uid, fields = None, context={})
```

```
def fields_view_get(self, cr, uid, view_id=None, view_type='form',context={})
```

```
def distinct_field_get(self, cr, uid, field, value, args=[], offset=0,limit=2000)
```

## 记录名字存取方法

- name\_get
- name\_search

```
def name_get(self, cr, uid, ids, context={})
```

```
def name_search(self, cr, uid, name="", args=[], operator='ilike',context={})
```

## 缺省值存取方法

- default\_get,
- default\_set

```
def name_get(self, cr, uid, ids, context={})
```

```
def name_search(self, cr, uid, name=, args=[], operator='ilike', context={})
```

## create 方法

在数据表中插入一条记录(或曰新建一个对象的 resource)。

格式:

```
def create(self, cr, uid, vals, context={})
```

参数说明:

**vals** : 待新建记录的字段值,是一个字典,形如: {'name\_of\_the\_field':value, ...}

**context (optional)** : OpenERP 几乎所有的方法都带有参数 context,context 是一个字典,存放一些上下文值,例如当前用户的信息,包括语言、角色等。

context 可以塞入任何值,在 action 定义中,有一个 context 属性,在界面定义时,可以在该属性中放入任何值,context 的最初值通常来自该属性值。 返回值:新建记录的 id。

举例:

```
id = pooler.get_pool(cr.dbname).get('res.partner.event').create(cr, uid,{'name': 'Email s
```

## search 方法

查询符合条件的记录。

格式:

```
def search(self, cr, uid, args, offset=0, limit=2000)
```

参数说明:

- args: 包含检索条件的 tuples 列表,格式为: [('name\_of\_the\_field', 'operator', value), ...]。 可用的 operators 有:

```
=, >, <, <=, >= in  
like, ilike child_of
```

更详细说明,参考《OpenERP 应用和开发基础》中的“域条件”有关章节。

- offset (optional): 偏移记录数,表示不返回检索结果的前 offset 条。
- limit (optional): 返回结果的最大记录数。 返回值:符合条件的记录的 id list。

## read 方法

返回记录的指定字段值列表。

格式:

```
def read(self, cr, uid, ids, fields=None, context={})
```

参数说明:

- ids: 待读取的记录 id 列表,形如[1,3,5,...]
- fields (optional): 待读取的字段值,不指定的话,读取所有字段。
- context (optional): 参见 create 方法。 返回值:返回读取结果的字典列表,形如 [{'name\_of\_the\_field': value, ...}, ...]

## browse 方法

浏览对象及其关联对象。从数据库中读取指定的记录,并生成对象返回。和 read 等方法不同,本方法不是返回简单的记录,而是返回对象。

返回的对象可以直接使用"."存取对象的字段和方法,形如"object.name\_of\_the\_field",关联字段(many2one 等),也可以通过关联字段直接访问“相邻”对象。例如:

```
addr_obj = self.pool.get('res.partner.address').browse(cr, uid, contact_id)
```

```
nom = addr_obj.name
compte = addr_obj.partner_id.bank
```

这段代码先从对象池中取得对象 `res.partner.address` ,调用它的方法 `browse`,取得 `id=contact_id` 的对象,然后直接用`res.partner.address`取得"name"字段以及关联对象 `partner` 的银行(`addr_obj.partner_id.bank`)。

格式:

```
def browse(self, cr, uid, select, offset=0, limit=2000)
```

参数说明:

- `select`: 待返回的对象 `id`,可以是一个 `id`,也可以是一个 `id` 列表。
- `offset` (optional): 参见 `search` 方法。
- `limit` (optional): 参见 `search` 方法。

返回值:返回对象或对象列表。 注意:本方法只能在 `Server` 上使用,由于效率等原因,不支持 `rpc` 等远程调用。

## write 方法

保存一个或几个记录的一个或几个字段。 格式:

```
def write(self, cr, uid, ids, vals, context={})
```

参数说明:

- `ids`: 待修改的记录的 `id` 列表。
- `vals`: 待保存的字段新值,是一个字典,形如:

```
{'name_of_the_field': value, ...}。
```

- `context` (optional): 参见 `create` 方法。
- 返回值:如果没有异常,返回 `True`,否则抛出异常。

举例:

```
self.pool.get('sale.order').write(cr, uid, ids, {'state':'cancel'})
```

## unlink 方法

删除一个或几个记录。

格式:

```
def unlink(self, cr, uid, ids)
```

参数说明:

- ids: 待删除的记录的 id 列表。
- 返回值: 如果没有异常, 返回 True, 否则抛出异常。

## default\_get 方法

复位一个或多个字段的缺省值。

格式:

```
def default_get(self, cr, uid, fields, form=None, reference=None)
```

参数说明:

- fields: 希望复位缺省值的字段列表。
- form (optional): 目前似乎未用(5.06 版)。
- reference (optional): 目前似乎未用(5.06 版)。
- 返回值: 字段缺省值, 是一个字典, 形如: {'field\_name': value, ... }。
- 举例:

```
self.pool.get('hr.analytic.timesheet').default_get(cr, uid, ['product_id', 'product_uo
```

```
def default_get(self, cr, uid, fields, context=None):
    data = super(curent_object, self).default_get(cr, uid, fields, context=context)
    data['field']=value
    return data
```

## default\_set 方法

重置字段的缺省值。

格式:

```
def default_set(self, cr, uid, field, value, for_user=False)
```

参数说明:

- field: 待修改缺省值的字段。
- value: 新的缺省值。
- for\_user (optional): 修改是否只对当前用户有效, 还是对所有用户有效, 缺省值是对所有用户有效。

- 返回值: True

## 菜单和动作

---

## 菜单

菜单定义格式:

```
<menuitem id="menuitem_id" name="Position/Of/The/Menu/Item/In/The/Tree" action="action_id"
</menuitem>
```

其中

- id:菜单ID,唯一标志该菜单,ID不可以重复。该字段是必须的。
- name:菜单显示名,即界面上看到的菜单名,该字段是必须的。可用符号"/"定义菜单的显示路径,如 name="Sales Management/Sales Order/Sales Order in Progress",表示销售管理模块下的菜单Sales Order下的子菜单Sales Order in Progress。如果指定了菜单路径,可以不定义parent字段(该字段定义菜单的上级菜单)。
- action:菜单的响应动作的ID,即点击该菜单,系统调用哪个响应动作(Action)。该字段不是必须的,不指定Action的菜单通常是父菜单。
- icon:菜单的图标,不是必须字段。默认icon是STOCK\_OPEN。

可用的icon有:

```
STOCK_ABOUT,
STOCK_ADD,
STOCK_APPLY,
STOCK_BOLD,
STOCK_CANCEL,
STOCK_CDROM,
STOCK_CLEAR,
STOCK_CLOSE,
STOCK_COLOR_PICKER,
STOCK_CONNECT,
STOCK_CONVERT,
STOCK_COPY,
STOCK_CUT,
STOCK_DELETE,
STOCK_DIALOG_AUTHENTICATION,
STOCK_DIALOG_ERROR,
STOCK_DIALOG_INFO,
STOCK_DIALOG_QUESTION,
STOCK_DIALOG_WARNING,
STOCK_DIRECTORY,
STOCK_DISCONNECT,
STOCK_DND,
STOCK_DND_MULTIPLE,
STOCK_EDIT,
STOCK_EXECUTE,
STOCK_FILE,
STOCK_FIND,
STOCK_FIND_AND_REPLACE,
STOCK_FLOPPY,
```

```
STOCK_GOTO_BOTTOM,
STOCK_GOTO_FIRST,
STOCK_GOTO_LAST,
STOCK_GOTO_TOP,
STOCK_GO_BACK,
STOCK_GO_DOWN,
STOCK_GO_FORWARD,
STOCK_GO_UP,
STOCK_HARDDISK,
STOCK_HELP,
STOCK_HOME,
STOCK_INDENT,
STOCK_INDEX,
STOCK_ITALIC,
STOCK_JUMP_TO,
STOCK_JUSTIFY_CENTER,
STOCK_JUSTIFY_FILL,
STOCK_JUSTIFY_LEFT,
STOCK_JUSTIFY_RIGHT, STOCK_MEDIA_FORWARD, STOCK_MEDIA_NEXT, STOCK_MEDIA_PAUSE,
STOCK_MEDIA_PLAY, STOCK_MEDIA_PREVIOUS, STOCK_MEDIA_RECORD, STOCK_MEDIA_REWIND, STOCK_MED
terp-crm, terp-mrp, terp-product, terp-purchase, terp-sale, terp-tools, terp-administrati
```

- groups:指定哪些权限组可以看见该菜单,示例 (groups="admin,user")
- sequence:菜单的显示序号,序号越小,菜单显示越靠上。默认值是10. \*

## 示例

在代码文件:server/bin/addons/sale/sale\_view.xml, 有如下菜单定义。

```
<menuitem name="Sales Management/Sales Order/Sales Order in Progress" id="menu_action_ord
```



# 动作

---

## 概述

---

### 概述

动作(Action)定义系统如何响应用户的操作。例如,用户登录系统,双击invoice, 点击菜单,点击按钮,等等。 这些用户操作,系统会调用相应的Action响应用户。

Action有多种类型:

- Window: 打开一个窗口
- Report: 打印一个报表,包括 Custom Report,RML Report
- Wizard: 调用Wizard
- Execute: 执行Server端的一个方法(method)
- Group: 动作组,即连续调用多个actions,形成“动作串”。 动作常用场合有:
  - 用户登录系统(User connection)
  - 用户点击菜单
  - 用户点击画面右边工具条上的Print或Action 中的链接。

## 事件例子

---

### 点击菜单

当用户点击菜单 “Operations > Partners > Partners Contact”, 系统将发生如下响应:

1. 在IR\_相关表中查找对应Action
2. 执行Action:如果Action的类型是Window,系统打开一个新窗口(List or Form),显示选定的对象记录。

### 用户登录

当用户登录系统,系统查找该用户的默认登录动作,为用户显示特定画面。

1. 查找user file 取得 ACTION\_ID
2. 执行Action

## 动作定义

- Action Name: 动作名称, 可以任意取名。
- Action Type: 动作类型, 总是 'ir.actions.act\_window'。
- View Ref: 该动作关联的视图, 即动作应显示的畫面。
- Object: 该动作关联的对象, 即动作应显示哪个对象的数据, 或者说应显示哪个数据表的数据。
- Source Object: 指定应在哪个对象的视图的右边工具条上显示本Action。

## view

The view describes how the edition form or the data tree/list appear on screen. The views can be of 'Form' or 'Tree' type, according to whether they represent a form for the edition or a list/tree for global data viewing. A form can be called by an action opening in 'Tree' mode. The form view is generally opened from the list mode (like if the user pushes on 'switch view').

## domain

This parameter allows you to regulate which resources are visible in a selected view.(restriction) For example, in the invoice case, you can define an action that opens a view that shows only invoices not paid. The domains are written in python; list of tuples. The tuples have three elements;

- the field on which the test must be done
- the operator used for the test (<, >, =, like)
- the tested value For example, if you want to obtain only 'Draft' invoice, use the following domain; `[('state','=','draft')]` In the case of a simple view, the domain define the resources which are the roots of the tree. The other resources, even if they are not from a part of the domain will be posted if the user develop the branches of the tree.

# window Action

Actions are explained in more detail in section “Administration Modules - Actions”. Here’s the template of an action XML record :

```
<record model="ir.actions.act_window" id="action_id_1">
  <field name="name">action.name</field>
  <field name="view_id" ref="view_id_1"/>
  <field name="domain">["list of 3-tuples (max 250 characters)"]</field> <field name="conte
  <field name="view_type">form|tree</field>
  <field name="view_mode">form,tree|tree,form|form|tree</field>
  <field name="usage">menu</field>
  <field name="target">new</field>
</record>
```

Where

- id is the identifier of the action in the table “ir.actions.act\_window”. It must be unique.
- name is the name of the action (mandatory).
- view\_id is the name of the view to display when the action is activated. If this field is not defined, the view of a kind (list or form) associated to the object res\_model with the highest priority field is used (if two views have the same priority, the first defined view of a kind is used).
- domain is a list of constraints used to refine the results of a selection, and hence to get less records displayed in the view. Constraints of the list are linked together with an AND clause : a record of the table will be displayed in the view only if all the constraints are satisfied.
- context is the context dictionary which will be visible in the view that will be opened when the action is activated. Context dictionaries are declared with the same syntax as Python dictionaries in the XML file. For more information about context dictionaries, see section “ The context Dictionary”.
- res\_model is the name of the object on which the action operates.
- view\_type is set to form when the action must open a new form view, and is set to tree when the action must open a new tree view.
- view\_mode is only considered if view\_type is form, and ignored otherwise. The four possibilities are :
  - form,tree : the view is first displayed as a form, the list view can be displayed by clicking the “alternate view button” ;
  - tree,form : the view is first displayed as a list, the form view can be displayed by clicking the “alternate view button” ;
  - form : the view is displayed as a form and there is no way to switch to list view ;
  - tree : the view is displayed as a list and there is no way to switch to form view. (version 5 introduced graph and calendar views)
- usage is used [+ TODO +]

- target the view will open in new window like wizard. They indicate at the user that he has to open a new window in a new 'tab'. Administration > Custom > Low Level > Base > Action > Window Actions

## example of actions

### Examples of actions

This action is declared in server/bin/addons/project/project\_view.xml.

```
<record model="ir.actions.act_window" id="open_view_my_project">
  <field name="name">project.project</field>
  <field name="res_model">project.project</field>
  <field name="view_type">tree</field>
  <field name="domain">[( 'parent_id', '=', False), ( 'manager', '=', uid)]</field> <field name
```

This action is declared in server/bin/addons/stock/stock\_view.xml.

```
<record model="ir.actions.act_window" id="action_picking_form">
  <field name="name">stock.picking</field>
  <field name="res_model">stock.picking</field>
  <field name="type">ir.actions.act_window</field>
  <field name="view_type">form</field>
  <field name="view_id" ref="view_picking_form"/>
  <field name="context">{'contact_display': 'partner'}</field>
</record>
```



## Wizard Action

---

Here's an example of a .XML file that declares a wizard.

```
<?xml version="1.0"?>
<terp>
<data>
<wizard string="Employee Info" model="hr.employee"
name="employee.info.wizard" id="wizard_employee_info"/> </data>
</terp>
```

A wizard is declared using a wizard tag. See “Add A New Wizard” for more information about wizard XML. also you can add wizard in menu using following xml entry

```
<?xml version="1.0"?>
<terp>
<data>
<wizard string="Employee Info" model="hr.employee" name="employee.info.wizard" id="wizard
</data>
</terp>
```

## Report Action

Report declaration Reports in Open ERP are explained in chapter “Reports Reporting”. Here’s an example of a XML file that declares a RML report :

```
<?xml version="1.0"?>
<terp>
<data>
<report id="sale_category_print" string="Sales Orders By Categories" model="sale.order"
name="sale_category.print" rml="sale_category/report/sale_category_report.rml" menu="True"
/>
</data>
</terp>
```

A report is declared using a report tag inside a “data” block. The different arguments of a report tag are :

- id : an identifier which must be unique.
- string : the text of the menu that calls the report (if any, see below).
- model : the Open ERP object on which the report will be rendered.
- rml : the .RML report model. Important Note : Path is relative to addons/ directory.
- menu : whether the report will be able to be called directly via the client or not. Setting menu to False is useful in case of reports called by wizards.
- auto : determines if the .RML file must be parsed using the default parser or not. Using a custom parser allows you to define additional functions to your report.

## 工作流

---

### 参考

<http://www.slideshare.net/AzizaMAhmed/work-flow-functional-concept-on-openerp7>

<http://www.withfan.com/blog/odooworkflow中的transitions/>

## workflow 定义

```
<?xml version="1.0"?>
<terp><data>
<record model="workflow" id=workflow_id>
<field name="name">workflow.name</field>
<field name="osv">resource.model</field>
<field name="on_create">True | False</field>
</record>
</data></terp>
```

- model: 固定取值 "workflow"
- id: 任意值, 唯一标识本 workflow
- name: workflow 的名称, 任意定义
- osv: 本 workflow 关联的对象类型, 是 OpenERP 模块中定义的某对象名, 如采购单对象 (purchase.order)。是本 workflow 处理的数据对象。
- on\_create: 每当系统新产生一个 osv 中定义的对象实例时候, 是否对应的产生一个和该对象实例关联的 workflow 实例。默认是 True。

## workflow 和 workflow 实例

workflow 定义了对某一类型的对象, 如采购订单 (PO) 的处理流程。例如, PO 单的一般处理流程也许是:

1. 新建 PO, State = draft;
2. 审批 PO, 审批的同时
  - i. 系统自动产生收货单, 工仓库收货;
  - ii. 系统自动产生凭据 (Invoice), 供财务确认付款;
  - iii. 系统自动产生 PDF 的采购订单, 并自动 EMail 给该 PO 单对应的供应商。但对于特定的某个 PO 对象, 需要一个 workflow 实例, 以记录本 PO 对象处在流程的哪个阶段, 如 PO1 尚在 draft 状态, PO2 已经审批通过。PO 单的审批, 以及对应的 a)、b)、c) 的动作, 都可以在 OE 的 workflow 中定义解决, 而不需要全编码在 PO 对象上。即 workflow 实现了流程处理相关的代码和被处理对象的代码相分离, 降低了不同处理代码的耦合性, 增加了系统功能的柔软性。

## 活动定义

```
<record model="workflow.activity" id="activity_id">
<field name="wkf_id" ref="workflow_id"/>
<field name="name">activity.name</field>
<field name="kind">dummy | function | subflow | stopall</field>
<field name="subflow_id">subflow_id</field>
<field name="action">(...)</field>
<field name="action_id">(...)</field>
<field name="split_mode">XOR | OR | AND</field>
<field name="join_mode">XOR | AND</field>
<field name="signal_send">(...)</field>
<field name="flow_start">True | False</field>
<field name="flow_stop">True | False</field>
</record>
```

- model:固定取值 workflow.activity
- wkf\_id:本 Activity 所属的工作流 id
- name: 本 Activity 名称,任意值
- kind:本 Activity 类型,有 Dummy, Function, Subflow, Stop All 四种。
  - kind 说明,如果流程到达本节点,系统应执行的动作类别。
  - Dummy 表示不执行任何动作,即 action 中定义的代码不会被执行。
  - Function 表示执行 action 中定义的 python 代码,且,执行 action\_id 中定义的 server action。常见情况是,action 中定义一个 write 方法,修改流程关联的对象的状态。对于 Function 类型的节点,action 中定义的代码或者返回 False,或者返回一个客户端动作 id(A client action should be returned)。
  - Subflow 类型表示触发“subflow\_id”中指定的工作流。仔细的读者或许要问,工作流的执行总是和某个被处理的对象关联,是的,如果定义了 action,subflow 关联的对象 id 由 action 中定义的代码返回。如果没有定义 action,系统默认 subflow 关联的对象和本节点所属的工作流处理的对象 id 一致。
  - stopall 类型表示,流程到此节点则结束,但结束前,系统仍会执行 action 中的代码。
- signal\_send: 执行完本节点的动作 (action 及 action\_id 定义的动作)后,应向别的工作流发往的 signal,格式是:subflow.signal。subflow\_id 和 signal\_send 必须配合使用,subflow\_id 表示,触发子工作流 subflow\_id,在该子工作流中,通常必须定义 signal\_send,signal\_send 定义父流程中的某个 signal,表示,子流程处理结束后触发父流程中的信号 subflow.signal。注意,用于父子流程通信的工作流 signal 必须是形如 subflow.\*。例如,在 HR 模块的 workflow "wkf\_expenses"中,需要开发票时候,它触发流程 account 模块中的工作流“account.wkf()”。account.wkf 处理完成后,发出信号 subflow.paid 通知 wkf\_expenses 流程 (subflow.paid)。wkf\_expenses 中定义了信号 subflow.paid (subflow.paid)。
- split\_mode: 有三个选项,XOR,OR,AND,默认是 XOR。
  - XOR 表示,由本节点始发的出迁移中,沿着第一个满足迁移条件的迁移跳转。

- OR 表示由本节点始发的出迁移中,只要满足迁移条件即沿该 迁移跳转。
- AND 表示由本节点始发的出迁移中,只有所有迁移皆满足迁移条件才跳转,而且是同时沿 所有迁移跳转。
- XOR 只有一个跳转,OR 有零或多个跳转,AND 有零或全部跳转。
- join\_mode:有两个选项,XOR,AND,默认是 XOR。
  - XOR 表示,以本节点为终点的入迁移中,只 要有一个跳至本节点,即执行本节点的 action。
  - AND 表示,以本节点为终点的入迁移中,只有所有迁 移都已经跳至本节点,才执行本节点的 action。
- flow\_start:表示流程的开始节点。
- flow\_stop:表示流程的结束节点。

## 迁移定义

迁移的完整 XML 定义格式如下。

```
<record model="workflow.transition" id="transition_id">
  <field name="act_from" ref="activity_id_1"/>
  <field name="act_to" ref="activity_id_2"/>
  <field name="group_id" ref="groupid"/>
  <field name="signal">(...)</field>
  <field name="condition">(...)</field>
  <field name="trigger_model">(...)</field>
  <field name="trigger_expr_id">(...)</field>
</record>
```

- act\_from:本迁移的起始节点,引用之前定义的 Activity。
- act\_to:本迁移的结束节点,引用之前定义的 Activity。
- group\_id:权限组,表示只有该权限组可以触发本迁移。
- signal: 触发本迁移的信号,表示,如果系统收到 signal 定义的信号,则触发本迁移。触发信号有三种方式,
  - 1)最常见的是用户点击视图中的“name = 本处定义的 signal”的 button,此时相当于向系统 发送迁移信号量。系统会根据视图中的对象 id,找到对象关联的 workflow,再找到与 button name 相同的 signal,触发之。
  - 2)调用 workflow\_service 的方法:trg\_validate(self, uid, res\_type, res\_id, signal, cr),此方法表示,触发对象类型 res\_type 关联的 workflow 的 signal 信号,工作流实例关联的 对象实例是 res\_id。
  - 3)子流程的 signal\_send 发出的信号,此种情况前文已说过。
- condition:迁移的条件,是一段 Python 代码,通常是一个函数调用。当系统收到 signal 中定义的信 号时,检查此处的条件,条件为真则实际触发迁移。
- trigger\_model 和 trigger\_expr\_id: 此二字段表示 启动一个新工作流实例。trigger\_model 定义对象类 型,trigger\_expr\_id 定义一段 Python 代码,返回 trigger\_model 类型的对象 id。此二字段表示,如果 act\_from 中的 action 执行完毕,且 condition 条件 OK,则系统中插入一个 trigger\_model 类型, trigger\_expr\_id 返回的对象 id 关联的工作流实例。然后,可以调用 workflow\_service 的方法 trg\_trigger(self, uid, res\_type, res\_id, cr)实际执行该工作流。实际使用例子请参考 Sale 模块的工作流 定义 wkf\_sale:

```
<field name="trigger_model">procurement.order</field>
<field name="trigger_expr_id">procurement_lines_get(</field>
```

## example

工作流程：

transition 相当过程， 连接两个activity, 每个activity 有自己的业务逻辑（做什么）

\_\_init\_\_.py

```
import test_model_workflow
```

\_\_openerp\_\_.py

```
# -*- coding: utf-8 -*-

{
    "name" : "workflow-sample",
    "description" : "Sample workflow to create a new instance based on another one's state",
    "version" : "0.1",
    "depends" : ["base"],
    "category": "Custom",
    "author" : "Matias Garcia Isaia",
    "url": "",
    "update_xml": ['test_workflow.xml'],
    "installable" : True,
    "active" : False
}
```

test\_model\_workflow.py

```
# -*- coding: utf-8 -*-
from openerp.osv import fields, osv

class test_model_end(osv.osv):
    _name = "test.model.end"
    _columns = { 'creado' : fields.char('Creado', size=64, required=False, readonly=False) }
    _defaults = { 'creado' : 'Por el aire' }
test_model_end()

class test_model_start(osv.osv):
    def create_other(self, cr, uids, ids):
        ends = self.pool.get('test.model.end')
        for start in self.browse(cr, uids, ids):
            ends.create(cr, uids, {'creado' : start.nombre })

    _name = "test.model.start"
    _columns = { 'crear_otro' : fields.boolean('Crear otro', required=False),
                  'nombre' : fields.char('Nombre', size=64, required=False, readonly=False) }
    _defaults = { 'crear_otro' : False,
                  'nombre' : 'Moron' }
```



```
test_model_start()
```

```
test_workflow.xml
```

```
<?xml version="1.0" ?>
<openerp>
  <data>
    <record model="workflow" id="workflow_test_mio">
      <field name="name">workflow.test</field>
      <field name="osv">test.model.start</field>
      <field name="on_create">True</field>
    </record>

    <record model="workflow.activity" id="activity_start_dummy">
      <field name="wkf_id" ref="workflow_test_mio" />
      <field name="name">workflow.test.activity.dummy</field>
      <field name="kind">dummy</field>
      <field name="flow_start">True</field>
    </record>

    <record model="workflow.activity" id="activity_test">
      <field name="wkf_id" ref="workflow_test_mio" />
      <field name="name">workflow.test.activity</field>
      <field name="kind">function</field>
      <field name="action">create_other()</field>
      <field name="flow_end">True</field>
    </record>

    <record model="workflow.transition" id="transition_test">
      <field name="act_from" ref="activity_start_dummy" />
      <field name="act_to" ref="activity_test" />
      <field name="condition">crear_otro == True</field>
      <!-- 接收的信号 -->
      <field name="signal">ahora_crealo</field>
    </record>

    <record model="ir.ui.view" id="test_model_start_form">
      <field name="name">test.model.start.form</field>
      <field name="model">test.model.start</field>
      <field name="type">form</field>
      <field name="arch" type="xml">
        <form string="Start">
          <field name="crear_otro" />
          <field name="nombre" />
          <!-- 触发信号 -->
          <button name="ahora_crealo" string="Dale duro Otto" />
        </form>
      </field>
    </record>

    <record model="ir.ui.view" id="test_model_start_tree">
      <field name="name">test.model.start.tree</field>
      <field name="model">test.model.start</field>
      <field name="type">tree</field>
      <field name="arch" type="xml">
```

```

        <tree string="Start">
            <field name="crear_otro" />
            <field name="nombre" />
        </tree>
    </field>
</record>

<record model="ir.actions.act_window" id="test_model_start_window">
    <field name="name">Test Model Start</field>
    <field name="res_model">test.model.start</field>
    <field name="view_type">form</field>
    <field name="view_mode">tree, form</field>
</record>

<record model="ir.ui.view" id="test_model_end_form">
    <field name="name">test.model.end.form</field>
    <field name="model">test.model.end</field>
    <field name="type">form</field>
    <field name="arch" type="xml">
        <form string="End">
            <field name="creado" />
        </form>
    </field>
</record>
<record model="ir.ui.view" id="test_model_end_tree">
    <field name="name">test.model.end.tree</field>
    <field name="model">test.model.end</field>
    <field name="type">tree</field>
    <field name="arch" type="xml">
        <tree string="End">
            <field name="creado" />
        </tree>
    </field>
</record>

<record model="ir.actions.act_window" id="test_model_end_window">
    <field name="name">Test Model End</field>
    <field name="res_model">test.model.end</field>
    <field name="view_type">form</field>
    <field name="view_mode">tree, form</field>
</record>

<menuitem id="test_model_start_menu" name="TEEEEST" action="test_model_start_windo
<menuitem id="test_model_end_menu" name="Finales" parent="test_model_start_menu"
</data>
</openerp>

```

## 报表开发

---

### 参考

<http://blog.emiprotechnologies.com/create-qweb-report-odoo/>

<https://www.odoo.com/documentation/8.0/reference/reports.html>

## 报表运行机制

1. OpenERP 报表的基本运行机制 OpenERP 报表的一般定义语法是:

```
<report id="c2c_demo_report_x" string="C2C Demo Report" model="hr.holidays" name="sandbox
```

这个定义的含义是,在对象 `hr.holidays` 上增加报表操作( `model="hr.holidays"` ),该报表操作的显示字符是 `C2C Demo Report(string="C2C Demo Report")`,当用户点击该操作字符(`C2C Demo Report`),系统调用名为 `sandbox_c2c_reporting_tools(name="sandbox_c2c_reporting_tools")` 的 Services,该 Services 返回报表文件(PDF 或其他格式文件)。因此,理解 OpenERP 报表机制的核心是,理解报表 Services 机制。

1. OpenERP 的报表 Service OpenERP 的报表 Service 的基本接口定义在文件:openerp-server-6.0.3\bin\report\interface.py, 期定义如下:

```
report_int(netsvc.Service)
__init__(self, name, audience='')
create(self, cr, uid, ids, datas, context=None)
```

init 方法中最重要的参数是 `name`,该参数是 Service Name,其格式是"report.xxx", xxx 必须和报表定义时候的(`name="sandbox_c2c_reporting_tools"`)一致,系统是通过该名字找到该 Service。

create 方法中,最重要的参数是 `ids`,该参数是报表操作所在的画面上,选定的对象的 id 列表。通常,系统会为 `ids` 中的每一个对象出一个报表。`datas` 参数通常用于 Wizard 的情况,即先弹出 Wizard 画面,用户输入一些数据,点击按钮,系统再输出报表文件。在这种情况下,`datas` 参数里保存着用户在 Wizard 画面上输入的数据。

显然,系统的内部动作是,用户点击报表动作,系统根据 `name="sandbox_c2c_reporting_tools"` 找到相应 Service,调用 Service 的 Create 方法,返回报表文件。Create 方法的返回值格式是:  
(`report_doc,mimetype`)。例如,如果返回 pdf 报表,返回值是(`pdf_doc,'pdf'`)。

1. RML 报表 如果直接继承接口 `report_int`,编写 create 方法生成 pdf 文档,代码复杂,工作量大。系统提供了 RML 格式报表,用于简化 pdf 报表开发。其基本原理是,开发 RML 格式文档,系统的 Create 方法读取 rml 文件,渲染成 pdf 文档,输出。相关接口如下:

```
report_rml(report_int)
__init__(self, name, table, tmpl, xsl)
create(self, cr, uid, ids, datas, context)
```

```
report_sxw(report_rml)
__init__(self, name, table, rml=False, parser=rml_parse, header='external', store=False)
create(self, cr, uid, ids, data, context=None)
```

这两个派生 Class 中,create 方法的参数没有变化,init 方法增加了一些参数,说明如下:

- table:报表关联的数据对象,渲染rml时候需要调用该对象取得数据。
- rml:RML文件路径及名称,系统需要读取该文件渲染成PDF报表。
- parser:渲染器,系统的实际做法是,在 create 方法中调用渲染器的有关方法,将 rml 渲染 成 pdf。用户可以开发自己的渲染器,用于将 rml 渲染成其他格式,如 html、txt 等,实际上,系统已经 提供了 html、txt 等的渲染器。因此,开发 rml 格式的报表时候,通常只需要开发自己的渲染器 (parser),不需要开发 report\_int。

## RML报表开发方法

---

# RML语法格式

---

# 权限设置

---



## From 7 to 8

---

## Backend Modules in V8

---

Raphael Collet (rco@odoo.com)

## Agenda

---

- Architecture of Odoo
- Module Open Academy

## Architecture of Odoo

---

### Architecture of Odoo

---

- Three-tier client/server/database
- Web client in Javascript
- Server and backend modules in Python
  - MVC framework

## Module Open Academy

---

### The Module

---

- Manage courses, sessions, and subscriptions
- Learn
  - Structure of a module
  - Definition of data models
  - Definition of views and menus

### Structure of a Module

---

An Odoo module is

```
* a python module (data models), with
* a manifest file,
* XML and CSV data files (base data, views, menus),
* frontend resources (Javascript, CSS).
```

## The Open Academy Module

---

The manifest file `__odoo__.py` ::

```
{
    'name': 'Open Academy',
    'version': '1.0',
    'category': 'Tools',
    'summary': 'Courses, Sessions, Subscriptions',
    'description': "...",
    'depends' : ['base'],
    'data' : ['view/menu.xml'],
    'images': [],
    'demo': [],
    'application': True,
}
```

## The Course Model

A model and its fields are defined in a Python class::

```
from odoo import Model, fields

class Course(Model):
    _name = 'openacademy.course'

    name = fields.Char(string='Title', required=True)
    description = fields.Text()
```

## The Menu as XML data

.. code-block:: xml

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

    <menuitem name="Open Academy" id="menu_root" sequence="110"/>

    <menuitem name="General" id="menu_general" parent="menu_root"/>

    <record model="ir.actions.act_window" id="action_courses">
      <field name="name">Courses</field>
      <field name="res_model">openacademy.course</field>
      <field name="view_mode">tree,form</field>
    </record>

    <menuitem name="Courses" id="menu_courses" parent="menu_general"
      sequence="1" action="action_courses"/>
  </data>
</openerp>
```

## Let's add a Form View

---

.. code-block:: xml

```
<record model="ir.ui.view" id="course_form">
  <field name="name">course form view</field>
  <field name="model">openacademy.course</field>
  <field name="arch" type="xml">

    <form string="Course" version="7.0">
      <sheet>
        <h1>
          <field name="name" placeholder="Course Title"/>
        </h1>
        <notebook>
          <page string="Description">
            <field name="description"/>
          </page>
        </notebook>
      </sheet>
    </form>

  </field>
</record>
```

## The Session Model

---

.. code::

```
class Session(Model):
    _name = 'openacademy.session'

    name = fields.Char(required=True)
    start_date = fields.Date()
    duration = fields.Integer(help="Duration in days")
    seats = fields.Integer(string="Number of Seats")
```

## Relational Fields

---

Let us link sessions to courses and instructors::

```
class Session(Model):
    _name = 'openacademy.session'

    ...

    course = fields.Many2one('openacademy.course', required=True)
    instructor = fields.Many2one('res.partner')
```

.. nextslide::

Let us back-link courses and sessions::

```
class Course(Model):
    _name = 'openacademy.course'

    ...

    responsible = fields.Many2one('res.users')
    sessions = fields.One2many('openacademy.session', 'course')
```

.. nextslide::

Let us link sessions to partners for attendee subscription::

```
class Session(Model):
    _name = 'openacademy.session'

    ...

    attendees = fields.Many2many('res.partner')
```

## Computed Fields

---

The value of those fields is computed::

```
class Session(Model):
    _name = 'openacademy.session'

    ...

    taken_seats = fields.Float(compute='_compute_taken_seats')

    @api.one
    @api.depends('attendees', 'seats')
    def _compute_taken_seats(self):
        if self.seats:
            self.taken_seats = 100.0 * len(self.attendees) / self.seats
        else:
            self.taken_seats = 0.0
```

## About self

---

Model instances are **recordsets**.

A recordset is an hybrid concept:

```
* collection of records
```

```
* record
```

.. code::

```
for session in self:
    print session.name
    print session.course.name

assert self.name == self[0].name
```

## Feedback with "Onchange" Methods

---

Modify form values when some field is filled in::

```
class Session(Model):
    _name = 'openacademy.session'

    ...

    @api.onchange('course')
    def _onchange_course(self):
        if not self.name:
            self.name = self.course.name
```

## Default Values

---

Specify the initial value to use in a form::

```
class Session(Model):
    _name = 'openacademy.session'

    ...

    active = fields.Boolean(default=True)
    start_date = fields.Date(default=fields.Date.today)

    ...
```

## Model Constraints

---

Prevent bad data::

```
from odoo.exceptions import Warning

class Session(Model):
    _name = 'openacademy.session'
```

```
...

@api.one
@api.constrains('instructor', 'attendees')
def _check_instructor(self):
    if self.instructor in self.attendees:
        raise Warning("Instructor of session '%s' "
                      "cannot attend its own session" % self.name)
```

## More Stuff

---

- Extend existing models
- Many view types
- Workflows
- Reports
- Security
- Translations

## Backend Modules in V8

---

## Conclusion

---

- Modules have a simple structure
- Model definition intuitive and efficient
  - uses Python standards (decorators, descriptors)
  - recordsets provide support for "batch" processing
  - many model hooks (default values, constraints, computed fields)