Contents

[1. Quick Start Guide 2](#_Toc365800699)

[2. Domain en OpenERP 3](#_Toc365800700)

[3. Attrs 4](#_Toc365800701)

[4. Domains in Filters 5](#_Toc365800702)

[5. Searching for null values in Conditions 7](#_Toc365800703)

[6. Special Tags in Views 8](#_Toc365800704)

[7. ORM Methods 13](#_Toc365800705)

# Quick Start Guide

**1°-Start reading:**

Quick start : [**Technical\_Memento\_v0.7.3**](https://www.openerp.com/files/memento/OpenERP_Technical_Memento_latest.pdf)

Official doc : [Modules OpenERP v7](http://doc.openerp.com/trunk/developers/server/03_module_dev/)

**2°-Sample module :**

Example 1 : [Basics of developing a simple module in OpenERP](http://zbeanztech.com/blog/basics-developing-simple-module-openerp)

Example 2 : [How to create a module in OpenERP ?](http://acespritechblog.wordpress.com/2012/05/22/how-to-create-a-module-in-openerp/)

Example 3: [Custom Sample Module Development](http://www.ibcscorp.com/application-integration-customization/erp/openerp-2/openerp-custom-module-development-quick-start-guide/)

Example 4:[Module structure of OpenERP module](http://nicechirag.blogspot.com/2012/01/openerp-complete-guide-for-module.html)

**3°- Videos**

[Create a module (through interface not with python code)](https://www.youtube.com/watch?v=RrXrxuAKFRY)

**Tools / IDE OpenERP**

**1°-IDE** : I suggest Eclipse IDE : [Eclipse Classic](http://www.eclipse.org/downloads/packages/eclipse-classic-422/junosr2)

**2°-Python** : plugin PyDev for eclipse : [Installing the Eclipse PyDev plugin](https://www.youtube.com/watch?v=ACrmcGjqrJw&wadsworth=1)

**3°-XML** : Editor integrated with eclipse

**4°-Templates Openerp** :[Template for eclipse of openerp](https://code.google.com/p/openerp-eclipse-template/) code snippets. This snippets are build to help rapid buildingopenerp project with great ease.

[For Python Snippets](http://www.youtube.com/watch?v=kfMk31VKxrY)

[For XML Snippets](http://www.youtube.com/watch?v=LCxYpofncKs)

**5°-OpenERP in eclipse**.

[Debug your openERP+python code in eclipse](https://www.youtube.com/watch?v=RTg8VvyEVak&wadsworth=1)

[OpenERP development environment in Windows using Eclipse.](http://www.zbeanztech.com/blog/openerp-source-eclipse-under-windows)

**6°-OpenERP V7.0 installation on ubuntu 12.04 x64**

And I want to end up this good post :

[How to install OpenERP V7.0 on Ubuntu 12.04 from launchpad repository?](http://help.openerp.com/question/2562/how-to-install-openerp-v70-on-ubuntu-1204-from-launchpad-repository/)

**\* Others links :**

* [Where is OpenERP v7 documentation?](http://help.openerp.com/question/6817/where-is-openerp-v7-documentation/)
* [How to create REPORTs in OpenERP...](http://sahotaparamjitsingh.blogspot.com/2012/04/how-to-create-reports-in-openerp.html)
* [create Kanban view in openerp 7.0](http://help.openerp.com/question/20113/create-kanban-view-in-openerp-70/#20119)

**Best start with openerp.**

# Domain en OpenERP

*Publicado por*[*Equipo de trey*](http://www.trey.es/author/Equipo%20de%20trey/)*el jun 23, 2010 en [OpenERP](http://www.trey.es/category/blog/openerp/" \o "Ver todas las entradas en OpenERP) |*[*1 comentario*](http://www.trey.es/blog/openerp/domain-en-openerp/#comments)

Los dominios son las condiciones que modifican el resultado de una consulta. Las expresiones son indicadas con un array, que contiene las condiciones, y en ocasiones, el operador booleano para unirlas. Veamos un ejemplo:

[ ('name', 'like', 'As'), '|', ('active', '=', True)]

Imaginemos que el anterior dominio lo aplicamos a un listado de clientes, el resultado será todos los clientes que contenga en su nombre ‘As’ **o** que la empresa esté activa.

Los operadores de las condiciones que podemos usar son:

* = , igual
* <> o != , distinto
* <= , menor o igual que…
* < , menor que…
* > , mayor que…
* >= , mayor o igual que…
* =like , contiene
* like , contiene
* not like , no contiene
* ilike , contiene (ignorando las mayúsculas y las minúsculas)
* not ilike , no contiene (ignorando las mayúsculas y las minúsculas)
* in , existe en un array …
* not in , no existe en un array…
* child\_of , es hijo de …

Los operadores booleanos permitidos son:

* & , Y
* | , O
* ! , NO

**DOMAINS:**

 [('user\_ids', 'in' , user.id)]

1. Each tuple in the search domain needs to have 3 elements, in the form: ('field\_name', 'operator', value), where:
2. **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.
3. **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.
4. **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))

# Attrs

As an OpenERP Developer, you might come across such a situation where you would like to make one or some fields readonly or mandatory or hide (‘invisible’ in OpenERP terminology) based on the values of other fields.

Well, then you are getting OpenERP Training just by reading this as the solutions here are explained by well versed OpenERP Trainers.

The solution here to your situation is ‘attrs’.

The attrs is an element in OpenERP which is responsible to alter the attributes of a field, not the value. The ‘attrs’ attribute can be used to dynamically change the attributes of view components based on the value of other fields. Buttons in OpenERP use states attribute for dynamic visibility, they too can be used with attrs. It can be used on field, page, group, button and notebook tags. Remember, there is no RPC call ever made because this is a pure client side attribute at view level.

The structure of ‘**attrs**‘ is a dictionary carrying the key as the client attribute and value as a set of domain(s),the keys could be:

* **readonly**
* **invisible**
* **required**

Structure : attrs=”{‘attribute’:Domain which is if true, the element will wear the new property ‘attribute’}”

Syntax of attrs : <field name=”value\_amount”

**attrs=”{‘readonly’:[('value','=','balance')],  ‘invisible’: ['|',('service','in',['OpenERP Support','OpenERP Training']),(‘company’,'=’,'SerpentCS’)]}”**

/>

Result : If the domains are satisfied, the field ‘value\_amount’ will be readonly in 1st case and will be hidden on 2nd case.

<field name=”tax\_code\_root\_id” **attrs=”{‘required’: [('parent\_id', '=', False)]}”**/>

 You can use any domains here but make sure there will be no server side calls made. So better to be limited at view level.

Attrs is reflected at view level per record, and hence that can **never** be applied at List view/ Tree view level.

# Domains in Filters

Some of you are maybe familiar with the use of <filter> and <field> in search views, this is about them. A few weeks ago the GTK client was changed to support a 'domain' attribute for <field> in search view, that works similarly to the 'domain' attribute of the <filter> elements.

In order to avoid a name clash and much confusion with the regular 'domain' attribute that exists for relational fields (such as many2one), we have renamed it to 'filter\_domain', referring to its behavior similar to the "domain" of a <filter>.

Example of a normal domain:

---------------------------

<search string="Search view of scrum.product.backlog">

<field name="project\_id" domain="[('scrum', '=', 1)]"/>

</search>

Effect: when using auto-completion for the project\_id m2o field, only projects with the 'scrum' attribute set to "1" would be suggested, and if using a selection widget, it will only contain scrum projects.

BTW, the domain at the view level (like above) is always combined with the domain specified in the model directly.

Example of a filter\_domain:

---------------------------

<field name="location\_id" context="{'location': self}"/>

<field name="location\_id" string="Location"

filter\_domain="['|',('location\_id','ilike',self),('location\_dest\_id','ilike',self)]"/>

</search>

Effect: when typing a value in the location\_id field of the search view, the filter\_domain is applied instead of the default domain [('location\_id','ilike', self)], so the filtering includes both stock moves that are going TO and stock moves that are coming FROM the given

locations.

You can of course combine a domain and a filter\_domain on the same field in a search view, but they will have 2 different effects.

fields.dummy and context fields

-------------------------------

Another use of the 'filter\_domain' attribute could be to disable the default behavior of a <field>, by setting filter\_domain="[]".

This could be convenient, for example if the field is supposed to be used to set the context, and not to filter, like the following:

<search string="Product search form with stock installed">

<field name="location\_id" context="{'location': self}"

filter\_domain="[]"/>

</search>

Most of the time you will not need this for context fields because these fields did not really exist on the object, and have been created as 'fields.dummy' (instead of 'fields.many2one' for example).

A fields.dummy field will not filter the results even if a domain is applied to it, so we don't really need the filter\_domain in this case.

# Searching for null values in Conditions

In python there is no null value, use False instead.

domain= [('mydatetime','=',False)]

# Special Tags in Views

**Header tag and status bar:**

All the buttons appear on the left side of the form and the status bar should appear on the right side of the form. And this has to be in the header tag as shown below...

<header>

<button name="create\_marklist" string="Create" type="object"/>

<field name="state" widget="statusbar"

  statusbar\_visible="draft,sent,invoiced,done"

  statusbar\_colors='{"draft":"red","done":"blue"}'/>

</header>

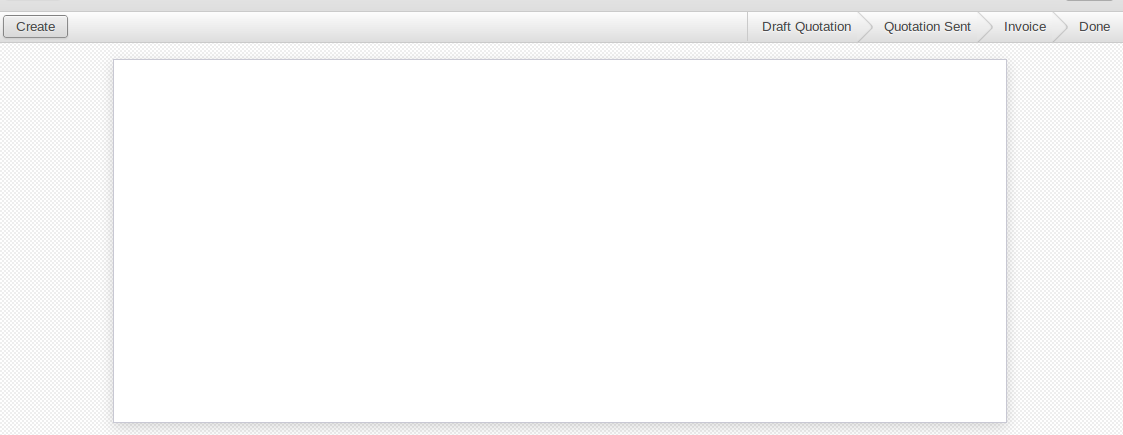
https://lh6.googleusercontent.com/OSJkPbrzvCMs-xI6pEP4Apq4ZFpF4B7qW3hyOYjKlGnBH7p1PmqiBro2gxGepTduvaINWnJC3arav84PDuIPJg3Pp-FyCDd2u6CfwtT0g7_sxVdgOdIiJ6V3

some new attributes have been added into the status bar . statusbar\_visible shows which all states to be visible to the user . To show the current status of the order statusbar\_colors have been used . Here when the status turns to invoice\_except the colour changes to red.

**Sheet tag:**

sheet tag gives your form a real document like appearance.

<sheet><sheet/>



**Heading tags:**

The <h1> to <h6> tags are used to define HTML headings.

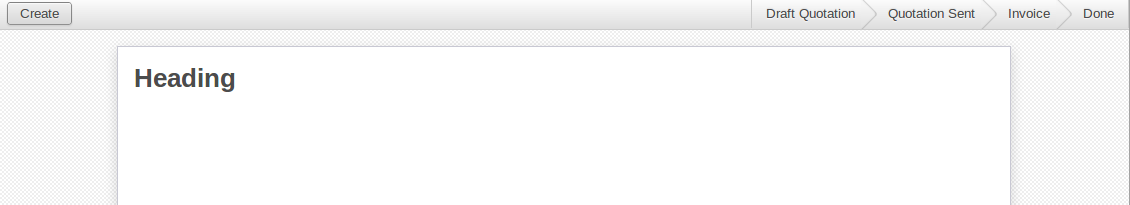
<h1> defines the most important heading. <h6> defines the least important heading.

So if you need to provide headings you can use the corresponding heading tag.

<h1>

  <label string="Heading " />

</h1>



**Label tag:**

Label tag is used to give labels to the corresponding fields. The for attribute tells which field should be provided with a label. An example is shown below

<label for="name"/>

<field name="name"/>



You can also use string attribute so that the label will change to what is given in the string.

<label String="Name"/>

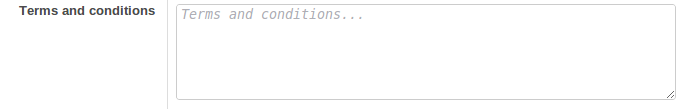
<field name="name"/>



**Placeholder attribute:**

If you need  a default string to appear in your text field you can provide an attribute placeholder as shown below.

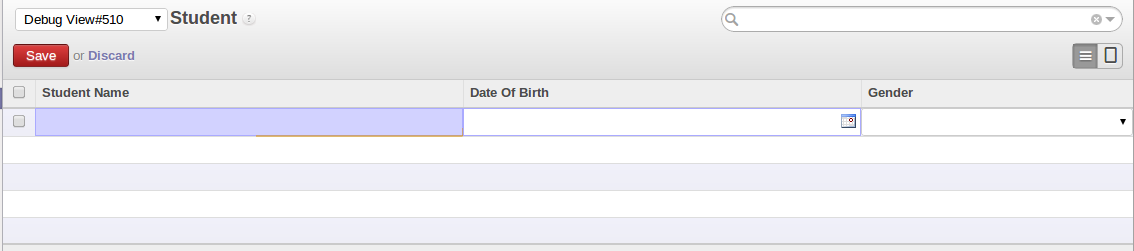
<field name="note" placeholder="Terms and conditions..."/>



**Editable attribute:**

To edit from the tree view itself use the editable attribute in your tree view. This makes the edition simpler.

<tree string="Student Details" editable="bottom">



**Help:**

Initialy in the tree view there will not  be any records . To direct the users to the next step a field called help is provided . You could provide the information using the paragraph tag <p></p> .

<field name="help" type="html">

    <p class="oe\_view\_nocontent\_create">

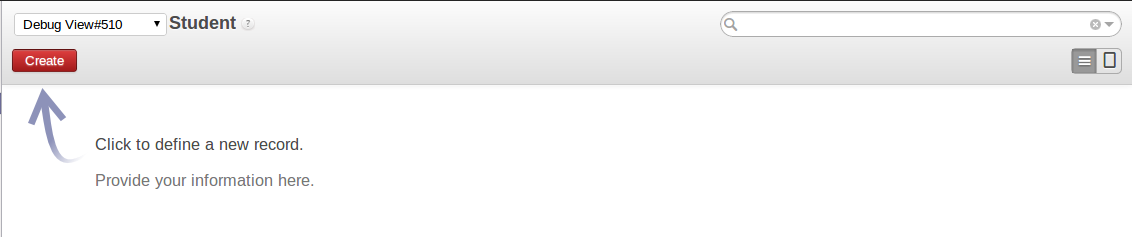
             Click to define a new record.

    </p><p>

            provide your information here.

    </p>

</field>



**Color and fonts attributes:**

You can also provide attributes in the tree string.

<tree string=”Sales order” fonts=”bold:message\_unread==True”

 colors="grey:state=='cancel';blue:state in (‘waiting\_date’,'manual');red:state in ('invoice','shipping')">

    <field name="order\_number"/>

    <field name="date\_order"/>

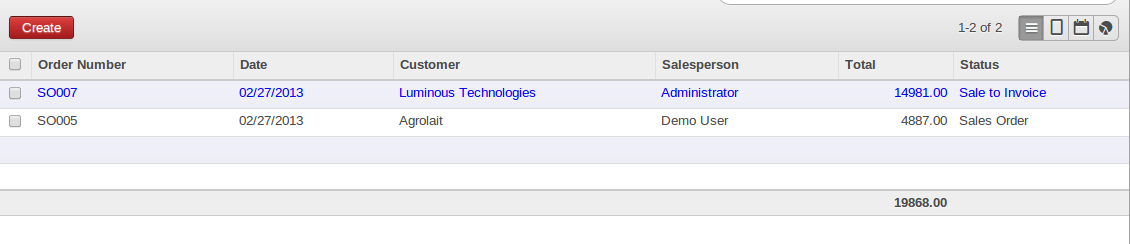
    <field name="partner\_id"/>

    <field name="user\_id"/>

    <field name="amount"/>

    <field name="state"/>

</tree>



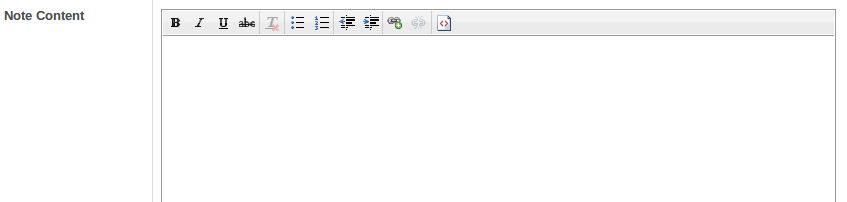
In the above example the font of the records appears bold and the colour of the records changes according to the states. If the state of the record is in manual state then the colour of the record in the tree view will be blue in colour as mentioned in the example.

**html fields:**

OpenERP 7.0 provides html fields too. An Example is shown below.

'memo': fields.html('Note Content'), (This is written in the python file)  
<field name="memo" widget="html" class="oe\_memo" editor\_height="450px" />

By providing this u can bring the editor in to your form



**Attachments:**

To attach files from your computer you need to use the ‘ir.attachment’ model. The field should have a many2many relation to the ‘ir.attachment’ model. An example is shown below.

'attachment\_ids': fields.many2many('ir.attachment', 'message','message\_id', 'attachment\_id', 'Attachments'),  
<field name="attachment\_ids"  widget="many2many\_binary"/>

https://lh4.googleusercontent.com/6hTpjza-z6gJZrFpZZD8-h91M07LK-0vwQCTeiCic5yHYUHeEEDiyvG_onY56Oh4DUp0aLqR1ly3DvE8dfBNuXoiv1pILLsW4QBFplHuCYwKztKvn1o9HalP

Widgets available in OpenERP 7.0

widget="one2many\_list" : Same as one2many in Openerp 7.0

widget="many2many\_tags" : same as many2many in Openerp 7.0

widget="monetary": You will be able to see  a dolar symbol after the value

widget="mail\_followers" : To add Followers

widget="mail\_thread" : Mail to groups

widget="statusbar" : shows the status bar

widget="progressbar" : Shows a progress bar

widget="html" : shows the html fields

widget="url" : Shows the url as a link

widget=”integer” : Only integer values will be saved  in the fields

widget="many2many\_kanban" :An add button appears and  you will be able to see the tree view of the corresponding model.

# ORM Methods

**Parmeters** used:

cr: database connection (cursor)

uid: id of user performing the operation

ids: list of record ids, or single integer when there is only one id

context: optional dictionary of contextual parameters, such as user language

**Methods**

**self.pool.get**('object\_name')

can be used to obtain a model class from anywhere

**create**(cr, uid, values, context=None)

Creates a new record with the specified value ,

Returns: id of the new record

**search**(cr, uid, args, offset=0, limit=None, order=None, context=None, count=False)

Returns: list of ids of records matching the given criteria

**read**(cr, user, ids, fields=None, context=None)

Returns: list of dictionaries with requested field values

**write**(cr, uid, ids, values, context=None)

Updates records with given ids with the given values.

Returns: True

**unlink**(cr, uid, ids, context=None)

Deletes records with the given ids .

Returns: True

**browse**(cr, uid, ids, context=None)

Fetches records as objects, allowing to use dot-notation to browse fields and relations

Returns: object or list of objects requested