Views

**Common Structure**

View objects expose a number of fields, they are optional unless specified otherwise.

name (mandatory)

only useful as a mnemonic/description of the view when looking for one in a list of some sort

model

the model linked to the view, if applicable (it doesn't for QWeb views)

priority

client programs can request views by id, or by (model, type). For the latter, all the views for the right type and model will be searched, and the one with the lowest priority number will be returned (it is the "default view").

priority also defines the order of application during [view inheritance](https://www.odoo.com/documentation/10.0/reference/views.html#reference-views-inheritance)

arch

the description of the view's layout

groups\_id

[Many2many](https://www.odoo.com/documentation/10.0/reference/orm.html#odoo.fields.Many2many) field to the groups allowed to view/use the current view

inherit\_id

the current view's parent view, see [Inheritance](https://www.odoo.com/documentation/10.0/reference/views.html#reference-views-inheritance), unset by default

mode

inheritance mode, see [Inheritance](https://www.odoo.com/documentation/10.0/reference/views.html#reference-views-inheritance). If inherit\_id is unset the mode can only be primary. If inherit\_id is set, extension by default but can be explicitly set to primary

application

website feature defining togglable views. By default, views are always applied

**Inheritance**

**View matching**

* if a view is requested by (model, type), the view with the right model and type, mode=primary and the lowest priority is matched
* when a view is requested by id, if its mode is not primary its *closest* parent with mode primary is matched

**View resolution**

Resolution generates the final arch for a requested/matched primary view:

1. if the view has a parent, the parent is fully resolved then the current view's inheritance specs are applied
2. if the view has no parent, its arch is used as-is
3. the current view's children with mode extension are looked up and their inheritance specs are applied depth-first (a child view is applied, then its children, then its siblings)

The result of applying children views yields the final arch

**Inheritance specs**

Inheritance specs are comprised of an element locator, to match the inherited element in the parent view, and children element that will be used to modify the inherited element.

There are three types of element locators for matching a target element:

* An xpath element with an expr attribute. expr is an [XPath](http://en.wikipedia.org/wiki/XPath) expression[2](https://www.odoo.com/documentation/10.0/reference/views.html#hasclass) applied to the current arch, the first node it finds is the match
* a field element with a name attribute, matches the first field with the same name. All other attributes are ignored during matching
* any other element: the first element with the same name and identical attributes (ignoring position and version attributes) is matched

The inheritance spec may have an optional position attribute specifying how the matched node should be altered:

inside (default)

the content of the inheritance spec is appended to the matched node

replace

the content of the inheritance spec replaces the matched node. Any text node containing only $0 within the contents of the spec will be replaced by a complete copy of the matched node, effectively wrapping the matched node.

after

the content of the inheritance spec is added to the matched node's parent, after the matched node

before

the content of the inheritance spec is added to the matched node's parent, before the matched node

attributes

the content of the inheritance spec should be attribute elements with a name attribute and an optional body:

* if the attribute element has a body, a new attributed named after its name is created on the matched node with the attribute element's text as value
* if the attribute element has no body, the attribute named after its name is removed from the matched node. If no such attribute exists, an error is raised

A view's specs are applied sequentially.

**Lists**

The root element of list views is <tree>[3](https://www.odoo.com/documentation/10.0/reference/views.html#treehistory). The list view's root can have the following attributes:

editable

by default, selecting a list view's row opens the corresponding [form view](https://www.odoo.com/documentation/10.0/reference/views.html#reference-views-form). The editable attributes makes the list view itself editable in-place.

Valid values are top and bottom, making *new* records appear respectively at the top or bottom of the list.

The architecture for the inline [form view](https://www.odoo.com/documentation/10.0/reference/views.html#reference-views-form) is derived from the list view. Most attributes valid on a [form view](https://www.odoo.com/documentation/10.0/reference/views.html#reference-views-form)'s fields and buttons are thus accepted by list views although they may not have any meaning if the list view is non-editable

default\_order

overrides the ordering of the view, replacing the model's default order. The value is a comma-separated list of fields, postfixed by desc to sort in reverse order:

<tree default\_order="sequence,name desc">

colors

Deprecated since version 9.0: replaced by decoration-{$name}

fonts

Deprecated since version 9.0: replaced by decoration-{$name}

decoration-{$name}

allow changing the style of a row's text based on the corresponding record's attributes.

Values are Python expressions. For each record, the expression is evaluated with the record's attributes as context values and if true, the corresponding style is applied to the row. Other context values are uid (the id of the current user) and current\_date (the current date as a string of the form yyyy-MM-dd).

{$name} can be bf (font-weight: bold), it (font-style: italic), or any [bootstrap contextual color](http://getbootstrap.com/components/#available-variations) (danger, info, muted, primary, success or warning).

create, edit, delete

allows *dis*abling the corresponding action in the view by setting the corresponding attribute to false

on\_write

only makes sense on an editable list. Should be the name of a method on the list's model. The method will be called with the id of a record after having created or edited that record (in database).

The method should return a list of ids of other records to load or update.

string

alternative translatable label for the view

Deprecated since version 8.0: not displayed anymore

Possible children elements of the list view are:

button

displays a button in a list cell

icon

icon to use to display the button

string

* if there is no icon, the button's text
* if there is an icon, alt text for the icon

type

type of button, indicates how it clicking it affects Odoo:

workflow (default)

sends a signal to a workflow. The button's name is the workflow signal, the row's record is passed as argument to the signal

object

call a method on the list's model. The button's name is the method, which is called with the current row's record id and the current context.

action

load an execute an ir.actions, the button's name is the database id of the action. The context is expanded with the list's model (as active\_model), the current row's record (active\_id) and all the records currently loaded in the list (active\_ids, may be just a subset of the database records matching the current search)

name

see type

args

see type

attrs

dynamic attributes based on record values.

A mapping of attributes to domains, domains are evaluated in the context of the current row's record, if True the corresponding attribute is set on the cell.

Possible attributes are invisible (hides the button) and readonly (disables the button but still shows it)

states

shorthand for invisible attrs: a list of states, comma separated, requires that the model has a state field and that it is used in the view.

Makes the button invisible if the record is *not* in one of the listed states

Danger

Using states in combination with attrs may lead to unexpected results as domains are combined with a logical AND.

context

merged into the view's context when performing the button's Odoo call

confirm

confirmation message to display (and for the user to accept) before performing the button's Odoo call

field

defines a column where the corresponding field should be displayed for each record. Can use the following attributes:

name

the name of the field to display in the current model. A given name can only be used once per view

string

the title of the field's column (by default, uses the string of the model's field)

invisible

fetches and stores the field, but doesn't display the column in the table. Necessary for fields which shouldn't be displayed but are used by e.g. @colors

groups

lists the groups which should be able to see the field

widget

alternate representations for a field's display. Possible list view values are:

progressbar

displays float fields as a progress bar.

many2onebutton

replaces the m2o field's value by a checkmark if the field is filled, and a cross if it is not

handle

for sequence fields, instead of displaying the field's value just displays a drag&drop icon

sum, avg

displays the corresponding aggregate at the bottom of the column. The aggregation is only computed on *currently displayed* records. The aggregation operation must match the corresponding field's group\_operator

attrs

dynamic attributes based on record values. Only effects the current field, so e.g. invisible will hide the field but leave the same field of other records visible, it will not hide the column itself

Note

if the list view is editable, any field attribute from the [form view](https://www.odoo.com/documentation/10.0/reference/views.html#reference-views-form) is also valid and will be used when setting up the inline form view

**Forms**

Form views are used to display the data from a single record. Their root element is <form>. They are composed of regular [HTML](http://en.wikipedia.org/wiki/HTML) with additional structural and semantic components.

**Structural components**

Structural components provide structure or "visual" features with little logic. They are used as elements or sets of elements in form views.

notebook

defines a tabbed section. Each tab is defined through a page child element. Pages can have the following attributes:

string (required)

the title of the tab

accesskey

an HTML [accesskey](http://www.w3.org/TR/html5/editing.html#the-accesskey-attribute)

attrs

standard dynamic attributes based on record values

group

used to define column layouts in forms. By default, groups define 2 columns and most direct children of groups take a single column. field direct children of groups display a label by default, and the label and the field itself have a colspan of 1 each.

The number of columns in a group can be customized using the col attribute, the number of columns taken by an element can be customized using colspan.

Children are laid out horizontally (tries to fill the next column before changing row).

Groups can have a string attribute, which is displayed as the group's title

newline

only useful within group elements, ends the current row early and immediately switches to a new row (without filling any remaining column beforehand)

separator

small horizontal spacing, with a string attribute behaves as a section title

sheet

can be used as a direct child to form for a narrower and more responsive form layout

header

combined with sheet, provides a full-width location above the sheet itself, generally used to display workflow buttons and status widgets

**Semantic components**

Semantic components tie into and allow interaction with the Odoo system. Available semantic components are:

button

call into the Odoo system, similar to [list view buttons](https://www.odoo.com/documentation/10.0/reference/views.html#reference-views-list-button)

field

renders (and allow edition of, possibly) a single field of the current record. Possible attributes are:

name (mandatory)

the name of the field to render

widget

fields have a default rendering based on their type (e.g. [Char](https://www.odoo.com/documentation/10.0/reference/orm.html#odoo.fields.Char), [Many2one](https://www.odoo.com/documentation/10.0/reference/orm.html#odoo.fields.Many2one)). The widget attributes allows using a different rendering method and context.

options

JSON object specifying configuration option for the field's widget (including default widgets)

class

HTML class to set on the generated element, common field classes are:

oe\_inline

prevent the usual line break following fields

oe\_left, oe\_right

[floats](https://developer.mozilla.org/en-US/docs/Web/CSS/float) the field to the corresponding direction

oe\_read\_only, oe\_edit\_only

only displays the field in the corresponding form mode

oe\_no\_button

avoids displaying the navigation button in a [Many2one](https://www.odoo.com/documentation/10.0/reference/orm.html#odoo.fields.Many2one)

oe\_avatar

for image fields, displays images as "avatar" (square, 90x90 maximum size, some image decorations)

groups

only displays the field for specific users

on\_change

calls the specified method when this field's value is edited, can generate update other fields or display warnings for the user

Deprecated since version 8.0: Use [odoo.api.onchange()](https://www.odoo.com/documentation/10.0/reference/orm.html#odoo.api.onchange) on the model

attrs

dynamic meta-parameters based on record values

domain

for relational fields only, filters to apply when displaying existing records for selection

context

for relational fields only, context to pass when fetching possible values

readonly

display the field in both readonly and edition mode, but never make it editable

required

generates an error and prevents saving the record if the field doesn't have a value

nolabel

don't automatically display the field's label, only makes sense if the field is a direct child of a group element

placeholder

help message to display in *empty* fields. Can replace field labels in complex forms. *Should not* be an example of data as users are liable to confuse placeholder text with filled fields

mode

for [One2many](https://www.odoo.com/documentation/10.0/reference/orm.html#odoo.fields.One2many), display mode (view type) to use for the field's linked records. One of tree, form, kanban or graph. The default is tree (a list display)

help

tooltip displayed for users when hovering the field or its label

filename

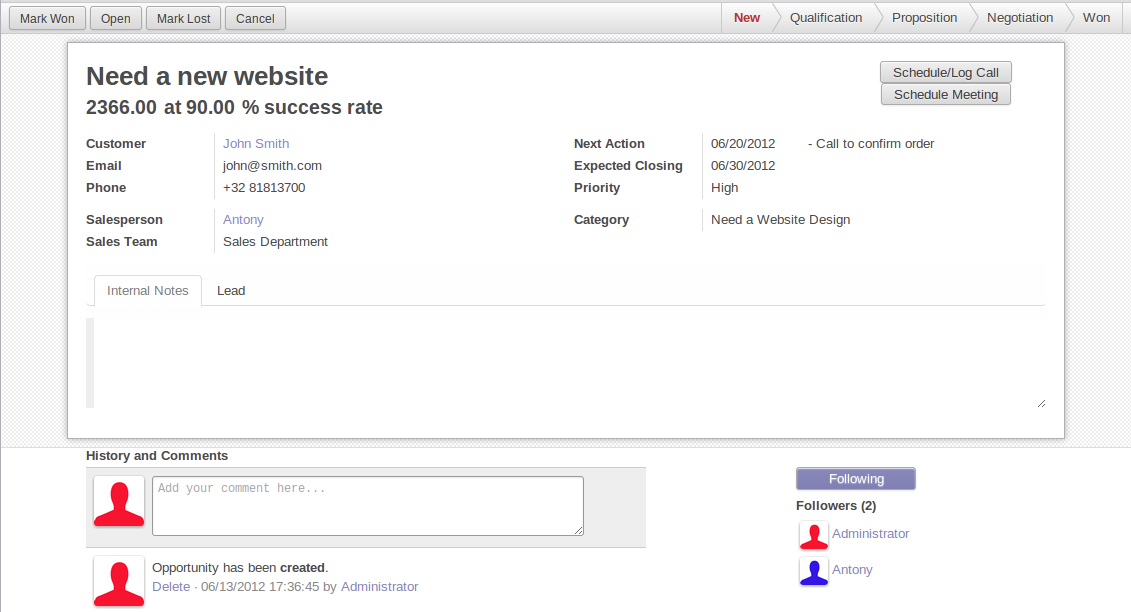
for binary fields, name of the related field providing the name of the file

password

indicates that a [Char](https://www.odoo.com/documentation/10.0/reference/orm.html#odoo.fields.Char) field stores a password and that its data shouldn't be displayed

**Business Views guidelines**

Business views are targeted at regular users, not advanced users. Examples are: Opportunities, Products, Partners, Tasks, Projects, etc.



In general, a business view is composed of

1. a status bar on top (with technical or business flow),
2. a sheet in the middle (the form itself),
3. a bottom part with History and Comments.

Technically, the new form views are structured as follows in XML:

<form>

<header> ... content of the status bar ... </header>

<sheet> ... content of the sheet ... </sheet>

<div class="oe\_chatter"> ... content of the bottom part ... </div>

</form>

**The Status Bar**

The purpose of the status bar is to show the status of the current record and the action buttons.

https://www.odoo.com/documentation/10.0/_images/status.png

**The Buttons**

The order of buttons follows the business flow. For instance, in a sale order, the logical steps are:

1. Send the quotation
2. Confirm the quotation
3. Create the final invoice
4. Send the goods

Highlighted buttons (in red by default) emphasize the logical next step, to help the user. It is usually the first active button. On the other hand, cancel buttons *must* remain grey (normal). For instance, in Invoice the button Refund must never be red.

Technically, buttons are highlighted by adding the class "oe\_highlight":

<button class="oe\_highlight" name="..." type="..." states="..."/>

**The Status**

Uses the statusbar widget, and shows the current state in red. States common to all flows (for instance, a sale order begins as a quotation, then we send it, then it becomes a full sale order, and finally it is done) should be visible at all times but exceptions or states depending on particular sub-flow should only be visible when current.

https://www.odoo.com/documentation/10.0/_images/status1.pnghttps://www.odoo.com/documentation/10.0/_images/status2.png

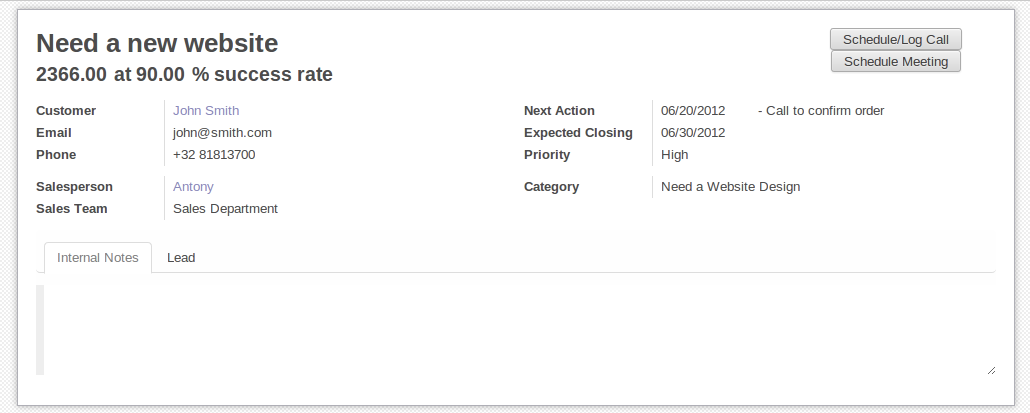
The states are shown following the order used in the field (the list in a selection field, etc). States that are always visible are specified with the attribute statusbar\_visible.

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

statusbar\_visible="draft,sent,progress,invoiced,done" />

**The Sheet**

All business views should look like a printed sheet:



1. Elements inside a <form> or <page> do not define groups, elements inside them are laid out according to normal HTML rules. They content can be explicitly grouped using <group> or regular <div> elements.
2. By default, the element <group> defines two columns inside, unless an attribute col="n" is used. The columns have the same width (1/n th of the group's width). Use a <group> element to produce a column of fields.
3. To give a title to a section, add a string attribute to a <group> element:
4. <group string="Time-sensitive operations">

this replaces the former use of <separator string="XXX"/>.

1. The <field> element does not produce a label, except as direct children of a <group> element[1](https://www.odoo.com/documentation/10.0/reference/views.html#backwards-compatibility). Use <label for="*field\_name*> to produce a label of a field.

**Sheet Headers**

Some sheets have headers with one or more fields, and the labels of those fields are only shown in edit mode.

| **View mode** | **Edit mode** |
| --- | --- |
| https://www.odoo.com/documentation/10.0/_images/header.png | https://www.odoo.com/documentation/10.0/_images/header2.png |

Use HTML text, <div>, <h1>, <h2>… to produce nice headers, and <label> with the class oe\_edit\_only to only display the field's label in edit mode. The class oe\_inline will make fields inline (instead of blocks): content following the field will be displayed on the same line rather than on the line below it. The form above is produced by the following XML:

<label for="name" class="oe\_edit\_only"/>

<h1><field name="name"/></h1>

<label for="planned\_revenue" class="oe\_edit\_only"/>

<h2>

<field name="planned\_revenue" class="oe\_inline"/>

<field name="company\_currency" class="oe\_inline oe\_edit\_only"/> at

<field name="probability" class="oe\_inline"/> % success rate

</h2>

**Button Box**

Many relevant actions or links can be displayed in the form. For example, in Opportunity form, the actions "Schedule a Call" and "Schedule a Meeting" have an important place in the use of the CRM. Instead of placing them in the "More" menu, put them directly in the sheet as buttons (on the top) to make them more visible and more easily accessible.

https://www.odoo.com/documentation/10.0/_images/header3.png

Technically, the buttons are placed inside a <div> to group them as a block on the top of the sheet.

<div class="oe\_button\_box" name="button\_box">

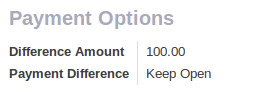
<button string="Schedule/Log Call" name="..." type="action"/>

<button string="Schedule Meeting" name="action\_makeMeeting" type="object"/>

</div>

**Groups and Titles**

A column of fields is now produced with a <group> element, with an optional title.



<group string="Payment Options">

<field name="writeoff\_amount"/>

<field name="payment\_option"/>

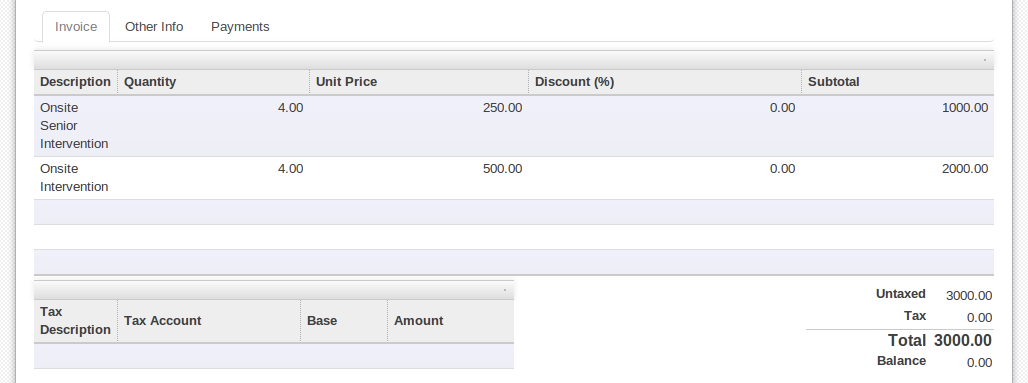
</group>

It is recommended to have two columns of fields on the form. For this, simply put the <group> elements that contain the fields inside a top-level <group> element.

To make [view extension](https://www.odoo.com/documentation/10.0/reference/views.html#reference-views-inheritance) simpler, it is recommended to put a name attribute on <group> elements, so new fields can easily be added at the right place.

**Special Case: Subtotals**

Some classes are defined to render subtotals like in invoice forms:



<group class="oe\_subtotal\_footer">

<field name="amount\_untaxed"/>

<field name="amount\_tax"/>

<field name="amount\_total" class="oe\_subtotal\_footer\_separator"/>

<field name="residual" style="margin-top: 10px"/>

</group>

**Placeholders and Inline Fields**

Sometimes field labels make the form too complex. One can omit field labels, and instead put a placeholder inside the field. The placeholder text is visible only when the field is empty. The placeholder should tell what to place inside the field, it *must not* be an example as they are often confused with filled data.

One can also group fields together by rendering them "inline" inside an explicit block element like <div>. This allows grouping semantically related fields as if they were a single (composite) fields.

The following example, taken from the *Leads* form, shows both placeholders and inline fields (zip and city).

| **Edit mode** | **View mode** |
| --- | --- |
| https://www.odoo.com/documentation/10.0/_images/placeholder.png | https://www.odoo.com/documentation/10.0/_images/screenshot-01.png |

<group>

<label for="street" string="Address"/>

<div>

<field name="street" placeholder="Street..."/>

<field name="street2"/>

<div>

<field name="zip" class="oe\_inline" placeholder="ZIP"/>

<field name="city" class="oe\_inline" placeholder="City"/>

</div>

<field name="state\_id" placeholder="State"/>

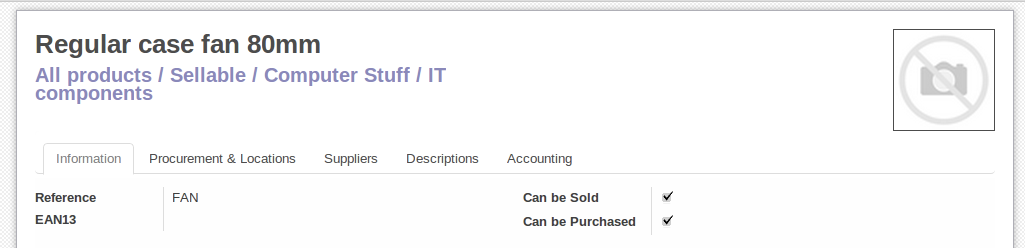
<field name="country\_id" placeholder="Country"/>

</div>

</group>

**Images**

Images, like avatars, should be displayed on the right of the sheet. The product form looks like:



The form above contains a <sheet> element that starts with:

<field name="product\_image" widget="image" class="oe\_avatar oe\_right"/>

**Tags**

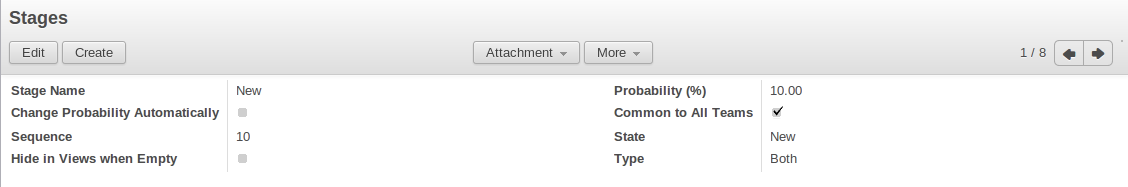
Most [Many2many](https://www.odoo.com/documentation/10.0/reference/orm.html#odoo.fields.Many2many) fields, like categories, are better rendered as a list of tags. Use the widget many2many\_tags for this:



<field name="category\_id" widget="many2many\_tags"/>

**Configuration forms guidelines**

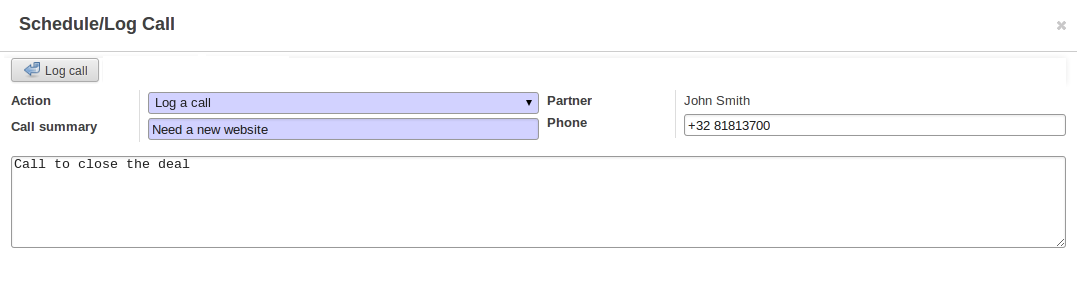
Examples of configuration forms: Stages, Leave Type, etc. This concerns all menu items under Configuration of each application (like Sales/Configuration).



1. no header (because no state, no workflow, no button)
2. no sheet

**Dialog forms guidelines**

Example: "Schedule a Call" from an opportunity.



1. avoid separators (the title is already in the popup title bar, so another separator is not relevant)
2. avoid cancel buttons (user generally close the popup window to get the same effect)
3. action buttons must be highlighted (red)
4. when there is a text area, use a placeholder instead of a label or a separator
5. like in regular form views, put buttons in the <header> element

**Configuration Wizards guidelines**

Example: Settings / Configuration / Sales.

1. always in line (no popup)
2. no sheet
3. keep the cancel button (users cannot close the window)
4. the button "Apply" must be red

**Graphs**

The graph view is used to visualize aggregations over a number of records or record groups. Its root element is <graph> which can take the following attributes:

type

one of bar (default), pie and line, the type of graph to use

stacked

only used for bar charts. If present and set to True, stacks bars within a group

The only allowed element within a graph view is field which can have the following attributes:

name (required)

the name of a field to use in a graph view. If used for grouping (rather than aggregating)

type

indicates whether the field should be used as a grouping criteria or as an aggregated value within a group. Possible values are:

row (default)

groups by the specified field. All graph types support at least one level of grouping, some may support more. For pivot views, each group gets its own row.

col

only used by pivot tables, creates column-wise groups

measure

field to aggregate within a group

interval

on date and datetime fields, groups by the specified interval (day, week, month, quarter or year) instead of grouping on the specific datetime (fixed second resolution) or date (fixed day resolution).

Warning

graph view aggregations are performed on database content, non-stored function fields can not be used in graph views

**Pivots**

The pivot view is used to visualize aggregations as a [pivot table](http://en.wikipedia.org/wiki/Pivot_table). Its root element is <pivot> which can take the following attributes:

disable\_linking

Set to True to remove table cell's links to list view.

display\_quantity

Set to true to display the Quantity column by default.

The elements allowed within a pivot view are the same as for the graph view.

**Kanban**

The kanban view is a [kanban board](http://en.wikipedia.org/wiki/Kanban_board) visualisation: it displays records as "cards", halfway between a [list view](https://www.odoo.com/documentation/10.0/reference/views.html#reference-views-list) and a non-editable [form view](https://www.odoo.com/documentation/10.0/reference/views.html#reference-views-form). Records may be grouped in columns for use in workflow visualisation or manipulation (e.g. tasks or work-progress management), or ungrouped (used simply to visualize records).

The root element of the Kanban view is <kanban>, it can use the following attributes:

default\_group\_by

whether the kanban view should be grouped if no grouping is specified via the action or the current search. Should be the name of the field to group by when no grouping is otherwise specified

default\_order

cards sorting order used if the user has not already sorted the records (via the list view)

class

adds HTML classes to the root HTML element of the Kanban view

group\_create

whether the "Add a new column" bar is visible or not. Default: true.

group\_delete

whether groups can be deleted via the context menu. Default: true.

group\_edit

whether groups can be edited via the context menu. Default: true.

quick\_create

whether it should be possible to create records without switching to the form view. By default, quick\_create is enabled when the Kanban view is grouped, and disabled when not.

Set to true to always enable it, and to false to always disable it.

Possible children of the view element are:

field

declares fields to use in kanban *logic*. If the field is simply displayed in the kanban view, it does not need to be pre-declared.

Possible attributes are:

name (required)

the name of the field to fetch

templates

defines a list of [QWeb](https://www.odoo.com/documentation/10.0/reference/qweb.html#reference-qweb) templates. Cards definition may be split into multiple templates for clarity, but kanban views *must* define at least one root template kanban-box, which will be rendered once for each record.

The kanban view uses mostly-standard [javascript qweb](https://www.odoo.com/documentation/10.0/reference/qweb.html#reference-qweb-javascript) and provides the following context variables:

instance

the current [Web Client](https://www.odoo.com/documentation/10.0/reference/javascript.html#reference-javascript-client) instance

widget

the current [KanbanRecord()](https://www.odoo.com/documentation/10.0/reference/views.html#KanbanRecord), can be used to fetch some meta-information. These methods are also available directly in the template context and don't need to be accessed via widget

record

an object with all the requested fields as its attributes. Each field has two attributes value and raw\_value, the former is formatted according to current user parameters, the latter is the direct value from a [read()](https://www.odoo.com/documentation/10.0/reference/orm.html#odoo.models.Model.read) (except for date and datetime fields that are [formatted according to user's locale](https://github.com/odoo/odoo/blob/a678bd4e/addons/web_kanban/static/src/js/kanban_record.js#L102))

formats

the web.formats() module to manipulate and convert values

read\_only\_mode

self-explanatory

**buttons and fields**

While most of the Kanban templates are standard [QWeb](https://www.odoo.com/documentation/10.0/reference/qweb.html#reference-qweb), the Kanban view processes field, button and a elements specially:

* by default fields are replaced by their formatted value, unless they match specific kanban view widgets
* buttons and links with a type attribute become perform Odoo-related operations rather than their standard HTML function. Possible types are:

action, object

standard behavior for [Odoo buttons](https://www.odoo.com/documentation/10.0/reference/views.html#reference-views-list-button), most attributes relevant to standard Odoo buttons can be used.

open

opens the card's record in the form view in read-only mode

edit

opens the card's record in the form view in editable mode

delete

deletes the card's record and removes the card

**Javascript API**

***class* KanbanRecord()**

[Widget()](https://www.odoo.com/documentation/10.0/reference/javascript.html#Widget) handling the rendering of a single record to a card. Available within its own rendering as widget in the template context.

**kanban\_color(*raw\_value*)**

Converts a color segmentation value to a kanban color class oe\_kanban\_color\_*color\_index*. The built-in CSS provides classes up to a color\_index of 9.

**kanban\_getcolor(*raw\_value*)**

Converts a color segmentation value to a color index (between 0 and 9 by default). Color segmentation values can be either numbers or strings.

**kanban\_image(*model, field, id[, cache][, options]*)**

Generates the URL to the specified field as an image access.

Arguments

* **model** (String) -- model hosting the image
* **field** (String) -- name of the field holding the image data
* **id** -- identifier of the record contaning the image to display
* **cache** (Number) -- caching duration (in seconds) of the browser default should be overridden. 0 disables caching entirely

Returns

an image URL

Warning

kanban\_text\_ellipsis has been removed in Odoo 9. CSS text-overflow should be used instead.

**Calendar**

Calendar views display records as events in a daily, weekly or monthly calendar. Their root element is <calendar>. Available attributes on the calendar view are:

date\_start (required)

name of the record's field holding the start date for the event

date\_stop

name of the record's field holding the end date for the event, if date\_stop is provided records become movable (via drag and drop) directly in the calendar

date\_delay

alternative to date\_stop, provides the duration of the event instead of its end date

color

name of a record field to use for *color segmentation*. Records in the same color segment are allocated the same highlight color in the calendar, colors are allocated semi-randomly.

event\_open\_popup

opens the event in a dialog instead of switching to the form view, disabled by default

quick\_add

enables quick-event creation on click: only asks the user for a name and tries to create a new event with just that and the clicked event time. Falls back to a full form dialog if the quick creation fails

display

format string for event display, field names should be within brackets [ and ]

all\_day

name of a boolean field on the record indicating whether the corresponding event is flagged as day-long (and duration is irrelevant)

mode

Default display mode when loading the calendar. Possible attributes are: day, week, month

**Gantt**

Gantt views appropriately display Gantt charts (for scheduling).

The root element of gantt views is <gantt/>, it has no children but can take the following attributes:

date\_start (required)

name of the field providing the start datetime of the event for each record.

date\_stop

name of the field providing the end duration of the event for each record. Can be replaced by date\_delay. One (and only one) of date\_stop and date\_delay must be provided.

If the field is False for a record, it's assumed to be a "point event" and the end date will be set to the start date

date\_delay

name of the field providing the duration of the event

duration\_unit

one of minute, hour (default), day, week, month, year

default\_group\_by

name of a field to group tasks by

type

gantt classic gantt view (default)

consolidate values of the first children are consolidated in the gantt's task

planning children are displayed in the gantt's task

consolidation

field name to display consolidation value in record cell

consolidation\_max

dictionary with the "group by" field as key and the maximum consolidation value that can be reached before displaying the cell in red (e.g. {"user\_id": 100})

Warning

The dictionnary definition must use double-quotes, {'user\_id': 100} is not a valid value

string

string to display next to the consolidation value, if not specified, the label of the consolidation field will be used

fold\_last\_level

If a value is set, the last grouping level is folded

round\_dnd\_dates

enables rounding the task's start and end dates to the nearest scale marks

drag\_resize

resizing of the tasks, default is true

**Diagram**

The diagram view can be used to display directed graphs of records. The root element is <diagram> and takes no attributes.

Possible children of the diagram view are:

node (required, 1)

Defines the nodes of the graph. Its attributes are:

object

the node's Odoo model

shape

conditional shape mapping similar to colors and fonts in [the list view](https://www.odoo.com/documentation/10.0/reference/views.html#reference-views-list). The only valid shape is rectangle (the default shape is an ellipsis)

bgcolor

same as shape, but conditionally maps a background color for nodes. The default background color is white, the only valid alternative is grey.

arrow (required, 1)

Defines the directed edges of the graph. Its attributes are:

object (required)

the edge's Odoo model

source (required)

[Many2one](https://www.odoo.com/documentation/10.0/reference/orm.html#odoo.fields.Many2one) field of the edge's model pointing to the edge's source node record

destination (required)

[Many2one](https://www.odoo.com/documentation/10.0/reference/orm.html#odoo.fields.Many2one) field of the edge's model pointing to the edge's destination node record

label

Python list of attributes (as quoted strings). The corresponding attributes's values will be concatenated and displayed as the edge's label

label

Explanatory note for the diagram, the string attribute defines the note's content. Each label is output as a paragraph in the diagram header, easily visible but without any special emphasis.

**Search**

Search views are a break from previous view types in that they don't display *content*: although they apply to a specific model, they are used to filter other view's content (generally aggregated views e.g. [Lists](https://www.odoo.com/documentation/10.0/reference/views.html#reference-views-list) or [Graphs](https://www.odoo.com/documentation/10.0/reference/views.html#reference-views-graph)). Beyond that difference in use case, they are defined the same way.

The root element of search views is <search>. It takes no attributes.

Possible children elements of the search view are:

field

fields define domains or contexts with user-provided values. When search domains are generated, field domains are composed with one another and with filters using **AND**.

Fields can have the following attributes:

name

the name of the field to filter on

string

the field's label

operator

by default, fields generate domains of the form [(*name*, *operator*, *provided\_value*)] where name is the field's name and provided\_value is the value provided by the user, possibly filtered or transformed (e.g. a user is expected to provide the *label* of a selection field's value, not the value itself).

The operator attribute allows overriding the default operator, which depends on the field's type (e.g. = for float fields but ilike for char fields)

filter\_domain

complete domain to use as the field's search domain, can use a self variable to inject the provided value in the custom domain. Can be used to generate significantly more flexible domains than operator alone (e.g. searches on multiple fields at once)

If both operator and filter\_domain are provided, filter\_domain takes precedence.

context

allows adding context keys, including the user-provided value (which as for domain is available as a self variable). By default, fields don't generate domains.

Note

the domain and context are inclusive and both are generated if a context is specified. To only generate context values, set filter\_domain to an empty list: filter\_domain="[]"

groups

make the field only available to specific users

widget

use specific search widget for the field (the only use case in standard Odoo 8.0 is a selection widget for [Many2one](https://www.odoo.com/documentation/10.0/reference/orm.html#odoo.fields.Many2one) fields)

domain

if the field can provide an auto-completion (e.g. [Many2one](https://www.odoo.com/documentation/10.0/reference/orm.html#odoo.fields.Many2one)), filters the possible completion results.

filter

a filter is a predefined toggle in the search view, it can only be enabled or disabled. Its main purposes are to add data to the search context (the context passed to the data view for searching/filtering), or to append new sections to the search filter.

Filters can have the following attributes:

string (required)

the label of the filter

domain

an Odoo [domain](https://www.odoo.com/documentation/10.0/reference/orm.html#reference-orm-domains), will be appended to the action's domain as part of the search domain

context

a Python dictionary, merged into the action's domain to generate the search domain

name

logical name for the filter, can be used to [enable it by default](https://www.odoo.com/documentation/10.0/reference/views.html#reference-views-search-defaults), can also be used as [inheritance hook](https://www.odoo.com/documentation/10.0/reference/views.html#reference-views-inheritance)

help

a longer explanatory text for the filter, may be displayed as a tooltip

groups

makes a filter only available to specific users

Tip

New in version 7.0.

Sequences of filters (without non-filters separating them) are treated as inclusively composited: they will be composed with OR rather than the usual AND, e.g.

<filter domain="[('state', '=', 'draft')]"/>

<filter domain="[('state', '=', 'done')]"/>

if both filters are selected, will select the records whose state is draft or done, but

<filter domain="[('state', '=', 'draft')]"/>

<separator/>

<filter domain="[('delay', '<', 15)]"/>

if both filters are selected, will select the records whose state is draft **and** delay is below 15.

separator

can be used to separates groups of filters in simple search views

group

can be used to separate groups of filters, more readable than separator in complex search views

**Search defaults**

Search fields and filters can be configured through the action's context using search\_default\_*name* keys. For fields, the value should be the value to set in the field, for filters it's a boolean value. For instance, assuming foo is a field and bar is a filter an action context of:

{

'search\_default\_foo': 'acro',

'search\_default\_bar': 1

}

will automatically enable the bar filter and search the foo field for *acro*.

**QWeb**

QWeb views are standard [QWeb](https://www.odoo.com/documentation/10.0/reference/qweb.html#reference-qweb) templates inside a view's arch. They don't have a specific root element.

A QWeb view can only contain a single template[4](https://www.odoo.com/documentation/10.0/reference/views.html#template-inherit), and the template's name *must* match the view's complete (including module name) [external id](https://www.odoo.com/documentation/10.0/glossary.html#term-external-id).

[template](https://www.odoo.com/documentation/10.0/reference/data.html#reference-data-template) should be used as a shortcut to define QWeb views.

[[1]](https://www.odoo.com/documentation/10.0/reference/views.html#id3) for backwards compatibility reasons

[[2]](https://www.odoo.com/documentation/10.0/reference/views.html#id1) an extension function is added for simpler matching in QWeb views: hasclass(\*classes) matches if the context node has all the specified classes

[[3]](https://www.odoo.com/documentation/10.0/reference/views.html#id2) for historical reasons, it has its origin in tree-type views later repurposed to a more table/list-type display

[[4]](https://www.odoo.com/documentation/10.0/reference/views.html#id4) or no template if it's an inherited view, then [it should only contain xpath elements](https://www.odoo.com/documentation/10.0/reference/views.html#reference-views-inheritance)