

---

# **Odoo development Documentation**

***Release master***

**IT-Projects LLC**

**Jul 19, 2019**



---

## Contents

---

<b>1 First steps</b>	<b>3</b>
<b>2 Module Development</b>	<b>5</b>
2.1 Docs and manifests . . . . .	5
2.2 Guidelines . . . . .	25
2.3 Odoo Python . . . . .	25
2.4 XML . . . . .	39
2.5 HTML . . . . .	42
2.6 CSS . . . . .	43
2.7 YAML . . . . .	43
2.8 Javascript . . . . .	44
2.9 Frontend . . . . .	45
2.10 Point of Sale (POS) . . . . .	46
2.11 Access . . . . .	63
2.12 Hooks . . . . .	70
2.13 Source Diving . . . . .	73
2.14 Lint . . . . .	75
2.15 Other . . . . .	77
<b>3 Debugging</b>	<b>81</b>
3.1 Terminal logs . . . . .	81
3.2 Browser's Console . . . . .	82
3.3 Sources tab at Browser's dev tools . . . . .	82
3.4 Network tab at Browser's dev tools . . . . .	82
3.5 QWeb . . . . .	83
3.6 Typical errors . . . . .	84
<b>4 Quality assurance</b>	<b>89</b>
4.1 Test automation . . . . .	89
4.2 Manual testing . . . . .	97
<b>5 Porting Modules</b>	<b>101</b>
5.1 Common Migration Helpers . . . . .	101
5.2 11.0- → 12.0+ . . . . .	102
5.3 10.0- → 11.0+, py2 → py3, shared Settings, sudo.get_param() . . . . .	103
5.4 9.0- → 10.0+, __openerp__.py → __manifest__.py . . . . .	104
5.5 7.0- → 8.0+, (cr, uid, ids, context) → self.env . . . . .	105

---

<b>6 User documentation</b>	<b>107</b>
6.1 Module releasing checklist . . . . .	107
6.2 static/description/index.html . . . . .	109
6.3 Screenshots tools . . . . .	113
6.4 Module description . . . . .	114
6.5 Contact us block . . . . .	114
6.6 JS Tour . . . . .	114
6.7 Preview module on App Store . . . . .	121
6.8 Image sizes . . . . .	123
<b>7 Git and Github</b>	<b>127</b>
7.1 Initial git & github configuration . . . . .	127
7.2 Porting . . . . .	128
7.3 Conflict resolving . . . . .	130
7.4 Multi Pull Request . . . . .	131
7.5 Cancel lame commit . . . . .	132
7.6 Pull request from console . . . . .	132
7.7 Check remote bundings . . . . .	132
7.8 Files relocation . . . . .	133
7.9 Git stash . . . . .	135
7.10 Update Git . . . . .	136
7.11 Squash commits into one . . . . .	136
7.12 Create branch from another's Pull Request . . . . .	137
<b>8 Continuous Integration</b>	<b>139</b>
8.1 Runbot . . . . .	139
8.2 Odoo Travis Tests . . . . .	141
8.3 Coverage . . . . .	141
<b>9 Odoo</b>	<b>143</b>
9.1 Models . . . . .	143
9.2 How to use Odoo . . . . .	153
<b>10 Odoo administration</b>	<b>163</b>
10.1 Odoo installation . . . . .	163
10.2 Longpolling . . . . .	169
10.3 About longpolling . . . . .	169
10.4 --workers . . . . .	170
10.5 --db_maxconn . . . . .	170
10.6 --max-cron-threads . . . . .	172
10.7 --addons-path . . . . .	172
10.8 --log-handler . . . . .	172
10.9 --db-filter . . . . .	173
10.10 --load . . . . .	174
10.11 PosBox . . . . .	174
<b>11 Continuous Delivery</b>	<b>179</b>
<b>12 Maintenance</b>	<b>181</b>
12.1 Data Migration . . . . .	181
<b>13 IDE</b>	<b>183</b>
13.1 Emacs . . . . .	183
13.2 PyCharm . . . . .	185
13.3 Tmux . . . . .	187

13.4 Visual Studio Code . . . . .	189
<b>14 Remote Development</b>	<b>193</b>
14.1 Usage . . . . .	193
14.2 Containers administration . . . . .	195
<b>15 Other</b>	<b>199</b>
15.1 RST format . . . . .	199
15.2 Adjust chromium window size script . . . . .	200



Join our telegram group: [https://t.me/odoo\\_talks](https://t.me/odoo_talks)



# CHAPTER 1

---

## First steps

---

- *Install odoo*
- take the course [Building a module](#)
- read the article [Source diving](#)
- *Configure git*
- read [Company rules \(For IT-Projects LLC employees only\)](#)
- Get tasks from your Guru!
- Fork repo, clone repo to your machine, make commits, push updates, create Pull Request



# CHAPTER 2

---

## Module Development

---

### 2.1 Docs and manifests

#### 2.1.1 Files

All files from this section ought to be fully<sup>\*0</sup> prepared **before** any other files in new module. It helps you to review requirements again before you start.

##### **README.rst**

- *Guidlines*
  - *OCA's README*
- *Demo*
  - *addons-dev*
- *HTML Description*
- *Usage instructions*
- *Changelog*
- *Tested on*

---

<sup>0</sup> The only exception could be made for lists of files in `__manifest__.py` (“`data`”, “`qweb`”, “`demo`” fields).

## Guidelines

```
.. image:: https://img.shields.io/badge/license-LGPL--3-blue.png
:target: https://www.gnu.org/licenses/lgpl
:alt: License: LGPL-3

=====
{MODULE_NAME}
=====

{Put some short introduction first.}

{Then add more detailed description, technical specifications, any other information
that could be interested for other developers. Don't forget that Usage instructions
is a separated and has to be located in doc/index.rst file.}

Credits
=====
Contributors
-----
* ` {DEVELOPER_NAME} <https://it-projects.info/team/{DEVELOPER_GITHUB_USERNAME}>` __

Sponsors
-----
* ` IT-Projects LLC <https://it-projects.info>` __

Maintainers
-----
* ` IT-Projects LLC <https://it-projects.info>` __

    To get a guaranteed support
    you are kindly requested to purchase the module
    at `odoo apps store <https://apps.odoo.com/apps/modules/{VERSION}/{TECHNICAL_
NAME}>` __.

    Thank you for understanding!

    ` IT-Projects Team <https://www.it-projects.info/team>` __

Further information
=====
Demo: http://runbot.it-projects.info/demo/{REPO_NAME}/{BRANCH}

HTML Description: https://apps.odoo.com/apps/modules/{VERSION}/{TECHNICAL_NAME}/

Usage instructions: `<doc/index.rst>` __

Changelog: `<doc/changelog.rst>` __

Notifications on updates: `via Atom <https://github.com/it-projects-llc/{REPO_NAME}/
commits/{BRANCH}/{TECHNICAL_NAME}.atom>` __, `by Email <https://blogtrottr.com/?
subscribe=https://github.com/it-projects-llc/{REPO_NAME}/commits/{BRANCH}/
{TECHNICAL_NAME}.atom>` __

Tested on Odoo {VERSION} {ODOO COMMIT SHA TO BE UPDATED}
```

## OCA's README

- <https://raw.githubusercontent.com/OCA/maintainer-tools/master/template/module/README.rst>

## Demo

Link to the runbot. Supported repo names are below. Change branche name if needed.

```
Demo: http://runbot.it-projects.info/demo/access-addons/10.0
Demo: http://runbot.it-projects.info/demo/addons-dev/misc-addons-10.0-some_feature
Demo: http://runbot.it-projects.info/demo/110n-addons/10.0
Demo: http://runbot.it-projects.info/demo/mail-addons/10.0
Demo: http://runbot.it-projects.info/demo/misc-addons/10.0
Demo: http://runbot.it-projects.info/demo/odoo-saas-tools/10.0
Demo: http://runbot.it-projects.info/demo/odoo-telegram/10.0
Demo: http://runbot.it-projects.info/demo/pos-addons/10.0
Demo: http://runbot.it-projects.info/demo/rental-addons/10.0
Demo: http://runbot.it-projects.info/demo/website-addons/10.0
```

## addons-dev

In most cases, if you work in addons-dev, you shall not use demo link to addons-dev (e.g. `http://runbot.it-projects.info/demo/addons-dev/misc-addons-10.0-some_feature`). Use a link for target repo instead (e.g. `http://runbot.it-projects.info/demo/misc-addons/10.0`). You can use links to addons-dev only if you know who will use it.

## HTML Description

Link to app store, e.g.

```
HTML Description: https://apps.odoo.com/apps/modules/10.0/web_debranding/
```

You have to prepare this link even if the module is not published yet, i.e. link returns 404 error.

## Usage instructions

- `doc/index.rst`

## Changelog

- `doc/changelog.rst`

## Tested on

```
Tested on Odoo 10.0 03bc8c5f9ac53a3349c1caac222f7619a632cccd8
```

commit sha can be found as following

```
cd /path/to/odoo  
git rev-parse HEAD
```

### doc/index.rst

```
=====  
{MODULE_NAME}  
=====  
  
Installation  
=====  
{Instruction about things to do before actual installation}  
  
* {OPTIONAL}`Activate longpolling <https://odoodevelopment.readthedocs.io/en/latest/admin/longpolling.html>`__  
* {Additional notes if any}  
* `Install <https://odoodevelopment.readthedocs.io/en/latest/odoo/usage/install-module.html>`__ this module in a usual way  
  
Configuration  
=====  
  
{Instruction how to configure the module before start to use it}  
  
* `Log in as SUPERUSER <https://odoodevelopment.readthedocs.io/en/latest/odoo/usage/login-as-superuser.html>`__  
* `Activate Developer Mode <https://odoodevelopment.readthedocs.io/en/latest/odoo/usage/debug-mode.html>`__  
* Open menu ``[[ {Menu} ]] >> {Submenu} >> {Subsubmenu}``  
* Click ``[{Button Name}]``  
  
Usage  
=====  
  
{Instruction for daily usage. It should describe how to check that module works. What  
shall user do and what would user get.}  
  
* Open menu ``[[ {Menu} ]]>> {Submenu} >> {Subsubmenu}``  
* Click ``[{Button Name}]``  
* RESULT: {what user gets, how the modules changes default behaviour}  
  
Uninstallation  
=====  
  
{Optional section for uninstallation notes. Delete it if you don't have notes for  
uninstallation.}
```

This description will be available at app store under *Documentation* tab. Example: [https://www.odoo.com/apps/modules/8.0/pos\\_multi\\_session/](https://www.odoo.com/apps/modules/8.0/pos_multi_session/)

### \_\_manifest\_\_.py (\_\_openerp\_\_.py)

- *Filename*
- *Template*
- *name*
- *summary*
- *category*
  - *Hidden*
- *version*
- *author*
  - *author in OCA*
- *website*
- *license*
- *external\_dependencies*

## Filename

- Use `__openerp__.py` for odoo 9.0 and earlier versions.
- Use `__manifest__.py` for odoo 10.0 and later versions, but don't rename filename on porting module from one version to another

## Template

Use example below as template. What are important here:

- order of attributes
- not used attributes are represented
- quote characters (" , " ")
- empty lines
- no description attribute
- price and currency attributes are commented-out if not used
- comma after last item in list (e.g. in 'depends' attribute)
- add new line symbol at the end of file (i.e. right after last })

```
# Copyright 2019 {DEVELOPER_NAME} <https://it-projects.info/team/{DEVELOPER_GITHUB_
↪USERNAME}>
# License LGPL-3.0 or later (https://www.gnu.org/licenses/lgpl.html).
{
    "name": "'''{MODULE_NAME}'''",
    "summary": "'''{SHORT_DESCRIPTION_OF_THE_MODULE}'''",
    "category": "{MODULE_CATEGORY}",
    # "live_test_url": "http://apps.it-projects.info/shop/product/DEMO-URL?version=
↪{ODOO_BRANCH}",
    "images": []
}
```

(continues on next page)

(continued from previous page)

```

"version": "{ODOO_BRANCH}.1.0.0",
"application": False,

"author": "IT-Projects LLC, {DEVELOPER_NAME}",
"support": "apps@it-projects.info",
"website": "https://apps.odoo.com/apps/modules/{VERSION}/{TECHNICAL_NAME}/",
"license": "LGPL-3",
# "price": 9.00,
# "currency": "EUR",

"depends": [
    "{DEPENDENCY1}",
    "{DEPENDENCY2}",
],
"external_dependencies": {"python": [], "bin": []},
"data": [
    "{FILE1}.xml",
    "{FILE2}.xml",
],
"demo": [
    "demo/{DEMOFILE1}.xml",
],
"qweb": [
    "static/src/xml/{QWEBFILE1}.xml",
],
"post_load": None,
"pre_init_hook": None,
"post_init_hook": None,
"uninstall_hook": None,

"auto_install": False,
"installable": True,

# "demo_title": "{MODULE_NAME}",
# "demo_addons": [
# ],
# "demo_addons_hidden": [
# ],
# "demo_url": "DEMO-URL",
# "demo_summary": "{SHORT_DESCRIPTION_OF_THE_MODULE}",
# "demo_images": [
#     "images/MAIN_IMAGE",
# ]
}
}

```



```

22 +      ],
23 +      "demo": [
24 +      ],
25 +      "installable": True,
26 +      "auto_install": False,
27 +  }

```

wrong

See also:

- OCA's template: [https://github.com/OCA/maintainer-tools/blob/master/template/module/\\_\\_openerp\\_\\_.py](https://github.com/OCA/maintainer-tools/blob/master/template/module/__openerp__.py)

### name

It must be non-technical name of the module

### summary

Short description of the module. E.g. you can describe here which problem is solved by the module. It could sound as a slogan.

### category

Categories from the list below are preferred.

- Accounting
- Discuss
- Document Management
- eCommerce
- Human Resources
- Industries
- Localization
- Manufacturing
- Marketing
- Point of Sale
- Productivity
- Project
- Purchases
- Sales
- Warehouse
- Website
- Extra Tools

### Hidden

For technical modules Hidden category can be used:

```
"category": "Hidden",
```

Such modules are excluded from search results on app store.

### version

*Note: whenever you change version, you have to add a record in [changelog.rst](#)*

The version number in the module manifest should be the Odoo major version (e.g. 8.0) followed by the module x.y.z version numbers. For example: 8.0.1.0.0 is expected for the first release of an 8.0 module.

The x.y.z version numbers follow the semantics *breaking.feature.fix*:

- x increments when the data model or the views had significant changes. Data migration might be needed, or depending modules might be affected.
- y increments when non-breaking new features are added. A module upgrade will probably be needed.
- z increments when bugfixes were made. Usually a server restart is needed for the fixes to be made available.

If applicable, breaking changes are expected to include instructions or scripts to perform migration on current installations.

If a module ported to different odoo versions (e.g. 8 and 9) and some update is added only to one version (e.g. 9), then version is changed as in example below:

- init
  - 8.0.1.0.0
  - 9.0.1.0.0
- feature added to 8.0 and ported to 9.0
  - 8.0.1.1.0
  - 9.0.1.1.0
- feature added to 9.0 only and not going to be ported to 8.0:
  - 8.0.1.1.0
  - 9.0.1.2.0
- fix made in 9.0 only and not going to be ported to 8.0:
  - 8.0.1.1.0
  - 9.0.1.2.1
- fix made in 8.0 and ported to 9.0
  - 8.0.1.2.2
  - 9.0.1.2.2

i.e. two module branches cannot have same versions with a different meaning

### author

Use company first and then developer(s):

```
"author": "IT-Projects LLC, Developer Name",
```

In the main, if module already exists and you make small updates/fixes, you should not add your name to authors.

## author in OCA

For OCA's repositories put company name first, then OCA. Developers are listed in README file:

```
"author": "IT-Projects LLC, Odoo Community Association (OCA)",
```

## website

Url to personal page at company's website (e.g. "<https://it-projects.info/team/yelizariev>")

## license

IT-Projects LLC uses following licences:

- "GPL-3" for odoo 8.0 and below
- "LGPL-3" for odoo 9.0 and above

For OCA's repositories use "AGPL-3".

## external\_dependencies

Check if some python library exists:

```
"external_dependencies": {"python" : ["openid"]}
```

Check if some system application exists:

```
"external_dependencies": {"bin" : ["libreoffice"]}
```

See also: *External dependencies in odoo*

## doc/changelog.rst

Changelog must answer a question: *What is changed, do I need those updates?*

---

**Note:** Don't use too technical description of the updates. For fixes, describe which error fixed, when the error happened, but without diving too much in technical details

---

## Template

Use this for new modules

```
`1.0.0`  
-----  
- **Init version**
```

## Guidelines

```
`2.0.0`  
-----  
- **New:** absolutely new way of ..  
  
`1.2.0`  
-----  
- **New:** new interface for ...  
- **Improvement:** better way to do ...  
  
`1.0.1`  
-----  
- **Fix:** issue about ...  
- **Fix:** another issue about ...  
  
`1.0.0`  
-----  
- **Init version**
```

## icon.png

File icon.png must be located at /static/description/icon.png

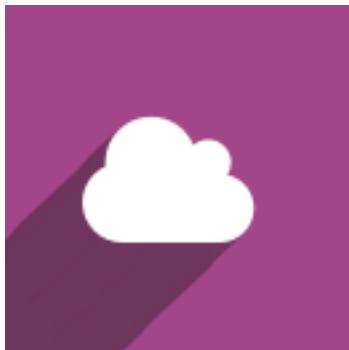
## IT-Projects LLC

Icons for IT-Projects LLC modules:

### TODO

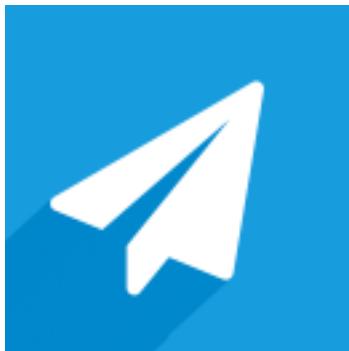
- *SaaS*
- *Telegram*
- *Access*
- *Barcode*
- *Mail*
- *Pos*
- *Stock*
- *Website*
- *Website\_Sale*
- *Misc*

SaaS



[Download](#)

Telegram



[Download](#)

Access



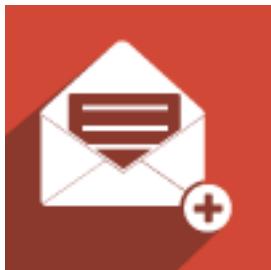
[Download](#)

**Barcode**



[Download](#)

**Mail**



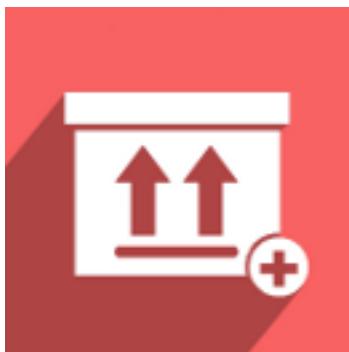
[Download](#)

**Pos**



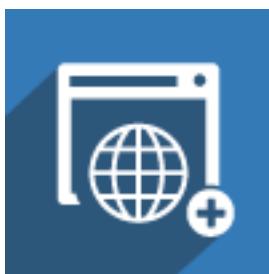
[Download](#)

**Stock**



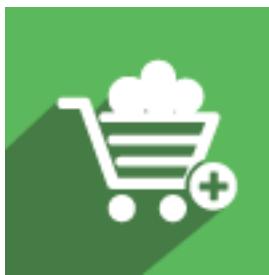
[Download](#)

**Website**



[Download](#)

**Website\_Sale**



[Download](#)

## Misc



[Download](#)

### 2.1.2 Notes

#### RST Requirements

Don't forget to keep correct rst format.

- *Extra lines*
- *References to menu*
- *Fields*
- *Checkboxes*
- *Buttons*
- *Selections*
- *Titles and sections*

#### Extra lines

Dont' forget about additional lines for correct formatting

#### Raw RST

```
This and next sentences are joined together.  
To split sentences to paragraphs you must add add empty line.
```

```
Spilted sentence 1.
```

```
Spilted sentence 2.
```

```
Lists below doesn't rendered correctly, because extra line is required:
```

```
* 1  
* 2  
* 3
```

(continues on next page)

(continued from previous page)

The same `for` sublist:

```
* 1
 * 1.1
 * 1.2
 * 1.3
* 2
```

Correctly formated lists:

```
* 1
* 2
* 3

 * 3.1
 * 3.2
 * 3.3

* 4
```

## Rendered RST

This and next sentences are joined together. To split sentences to paragraphs you must add add empty line.

Spited sentence 1.

Spited sentence 2.

Lists below doesn't rendered correctly, because extra line is required: \* 1 \* 2 \* 3

The same for sublist:

- 1 \* 1.1 \* 1.2 \* 1.3
- 2

Correctly formated lists:

- 1
- 2
- 3
  - 3.1
  - 3.2
  - 3.3
- 4

## References to menu

For menus use double back-quotes with **spaced** slash and with top menu surrounded by double square brackets :

```
OK:  
* Open menu ``[[ Settings ]] >> Parameters >> System Parameters``  
  
BAD  
* Open menu ``[[Settings]]>>Parameters>>System Parameters``  
* Open menu "[[ Settings ]] >> Parameters >> System Parameters"  
* Open menu ''[[ Settings ]] >> Parameters >> System Parameters''  
* Open menu ``[[ Settings ]] > Parameters > System Parameters``  
* Open menu ``[[ Settings ]]>> Parameters >> System Parameters``
```

## Fields

Use bold format for fields:

```
* Set **Name** and **Date** values
```

## Checkboxes

Same as Fields but draw box (with mark or without), e.g.:

```
* Set **[x] Use Longpooling**  
* Switch **[ ] Use Longpooling** off
```

## Buttons

Use square brackets in double back-quotes to name buttons. Keep letter cases the same as in UI.

```
OK:  
* click ``[Save]``  
  
Bad:  
* click ``[save]``
```

## Selections

Use arrow symbol -> to specify value in selection and many2one fields:

```
* Choose ``Partner -> Administrator``
```

## Titles and sections

```
OK:  
=====  
Correctly formatted Title  
=====  
  
Correctly formatted section  
=====
```

(continues on next page)

(continued from previous page)

**BAD:**

```
=====
No spaces at the beggining and end of title
=====

=====
No space at the end of title
=====

=====
Incorrect number of signs in title
=====

=====
Incorrect number of signs in title
=====

=====
Incorrect number of signs in section
=====

=====
Incorrect number of signs in section
=====
```

**Difference of doc files****README.rst**

Contains information interested for developers:

- short description
- technical details

**doc/index.rst**

Usage instruction. Used by end users after purchasing the module. It shall give an answer to the question “*How to check that module works (how to install, how to configure, how to use)?*”. Also, it may cover the question “*How to safely uninstall the module*”.

**static/description/index.html**

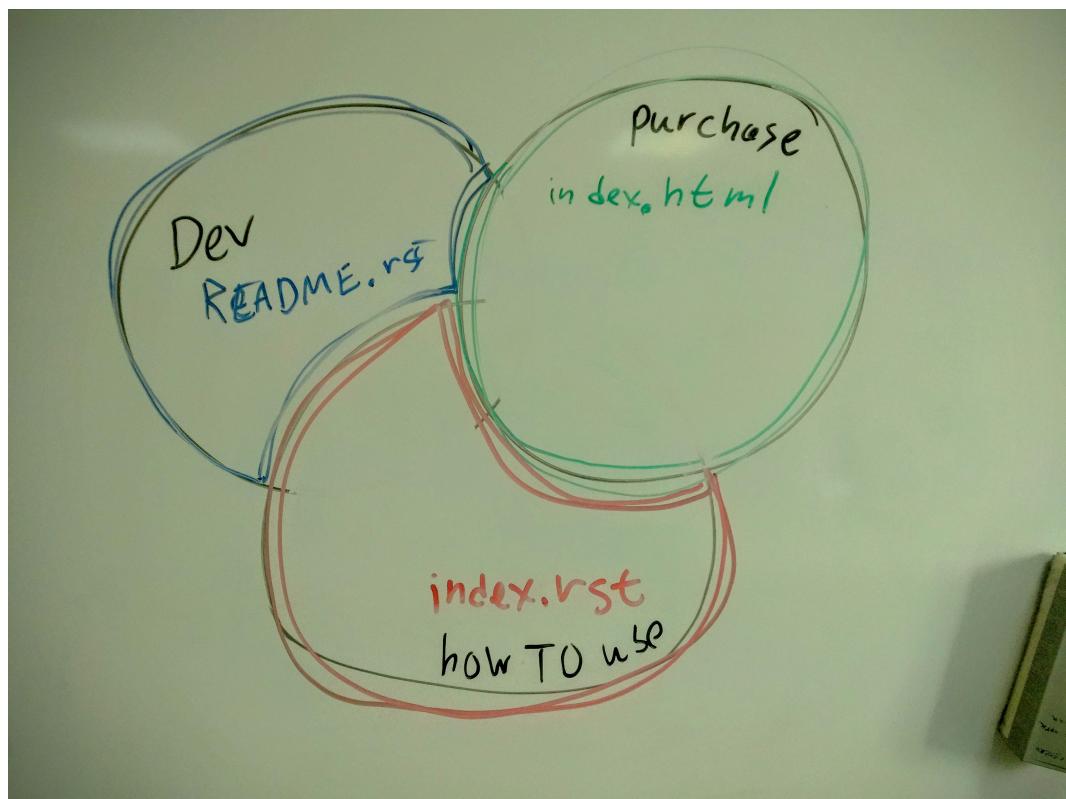
Module presentation. It shall give an answer to the question “*Is this module interesting for me?*”. Presentation has to give the answer as quickly as possible.

**Content intersection**

While every file has its own purpose, the content may intersect. If you don’t want duplicate content, use the following priority:

- index.html

- index.rst
- README.rst



### 2.1.3 Template handling

Download templates:

```
cd PATH/TO/MODULE-ROOT/  
  
# __manifest__.py  
wget -q https://raw.githubusercontent.com/it-projectsl1c/odoo-development/master/  
    ↵docs/dev/docs/templates/__manifest__.py  
# __README__.rst  
wget -q https://raw.githubusercontent.com/it-projectsl1c/odoo-development/master/  
    ↵docs/dev/docs/templates/README.rst  
mkdir doc  
cd doc  
# doc/index.rst  
wget -q https://raw.githubusercontent.com/it-projectsl1c/odoo-development/master/  
    ↵docs/dev/docs/templates/doc/index.rst  
# doc/changelog.rst  
wget -q https://raw.githubusercontent.com/it-projectsl1c/odoo-development/master/  
    ↵docs/dev/docs/templates/doc/changelog.rst  
cd ..  
# new __init__.py  
echo "# License LGPL-3.0 or later (https://www.gnu.org/licenses/lgpl.html)." >> __  
    ↵init__.py
```

(continues on next page)

(continued from previous page)

```
# OTHER TEMPLATES

# security/ir.model.access.csv
mkdir security
echo "id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink" >> \
→ security/ir.model.access.csv

# controllers/main.py
mkdir controllers
echo "from . import controllers" >> __init__.py
echo "# License LGPL-3.0 or later (https://www.gnu.org/licenses/lgpl.html). " >> \
→ controllers/__init__.py
echo "from . import main" >> controllers/__init__.py
echo "# Copyright 2019 {DEVELOPER_NAME} <https://it-projects.info/team/{DEVELOPER\_GITHUB\_USERNAME}>" >> controllers/main.py
echo "# License LGPL-3.0 or later (https://www.gnu.org/licenses/lgpl.html). " >> \
→ controllers/main.py

# only for 10.0- versions:
echo "# -*- coding: utf-8 -*-" >> controllers/main.py

#
```

Update templates:

```
# SETTINGS
# {braces} AND text inside them must be replaced to appropriate value (without braces)

# set your name
# you can add it to your ~/.bashrc, e.g.
# export DEVELOPER_NAME="Ivan Yelizariev"
# export DEVELOPER_GITHUB_USERNAME=yelizariev
DEVELOPER_NAME="{Ivan Yelizariev}"
DEVELOPER_GITHUB_USERNAME={yelizariev}

# this command returns name of current folder, so you MUST be at module's root
TECHNICAL_NAME=`basename $PWD` 

# module description
MODULE_NAME="{SOME Non-technical name}"
MODULE_SUMMARY="{SHORT module description for README and manifest}"

# Repository: choose one of the options
REPO_NAME=access-addons
REPO_NAME=l10n-addons
REPO_NAME=mail-addons
REPO_NAME=misc-addons
REPO_NAME=odoo-saas-tools
REPO_NAME=saas-addons
REPO_NAME=odoo-telegram
REPO_NAME=pos-addons
REPO_NAME=website-addons

# Branch: choose one of the options
```

(continues on next page)

(continued from previous page)

```

ODOO_BRANCH=12.0
ODOO_BRANCH=11.0
ODOO_BRANCH=10.0
ODOO_BRANCH=9.0
ODOO_BRANCH=8.0

# to get commit sha use following inside odoo repo: "git show HEAD | head"
ODOO_REVISION={ODOO_COMMIT_SHA_TO_BE_UPDATED}
# alternatively (use appropriate path to odoo source):
git -C ~/odoo/odoo-{$ODOO_BRANCH}/odoo fetch upstream && export ODOO_REVISION=`git -
→C ~/odoo/odoo-{$ODOO_BRANCH}/odoo rev-parse upstream/{$ODOO_BRANCH}`

# Category: choose one of the options
MODULE_CATEGORY="Accounting"
MODULE_CATEGORY="Discuss"
MODULE_CATEGORY="Document Management"
MODULE_CATEGORY="eCommerce"
MODULE_CATEGORY="Human Resources"
MODULE_CATEGORY="Industries"
MODULE_CATEGORY="Localization"
MODULE_CATEGORY="Manufacturing"
MODULE_CATEGORY="Marketing"
MODULE_CATEGORY="Point of Sale"
MODULE_CATEGORY="Productivity"
MODULE_CATEGORY="Project"
MODULE_CATEGORY="Purchases"
MODULE_CATEGORY="Sales"
MODULE_CATEGORY="Warehouse"
MODULE_CATEGORY="Website"
MODULE_CATEGORY="Extra Tools"
MODULE_CATEGORY="SaaS"
MODULE_CATEGORY="Hidden"

# icon: choose one of options
ICON=access
ICON=barcode
ICON=mail
ICON=misc
ICON=pos
ICON=saaS
ICON=stock
ICON=telegram
ICON=website
ICON=website_sale

# EXECUTING
mkdir -p static/description
# static/description/icon.png
wget -q https://raw.githubusercontent.com/it-project-llc/odoo-development/master/
→docs/images/module-icons/${ICON}/icon.png -O static/description/icon.png

sed -i "s/{MODULE_NAME}/${MODULE_NAME}/g" __manifest__.py README.rst doc/index.rst
sed -i "s/{Put some short introduction first.}/${MODULE_SUMMARY}/g" README.rst
sed -i "s/{SHORT_DESCRIPTION_OF_THE_MODULE}/${MODULE_SUMMARY}/g" __manifest__.py
sed -i "s/{MODULE_CATEGORY}/${MODULE_CATEGORY}/g" __manifest__.py
sed -i "s/{DEVELOPER_NAME}/${DEVELOPER_NAME}/g" __manifest__.py README.rst doc/index.
→rst controllers/main.py

```

(continues on next page)

(continued from previous page)

```

sed -i "s/{DEVELOPER_GITHUB_USERNAME}/${DEVELOPER_GITHUB_USERNAME}/g" __manifest__.py
sed -i "s/README.rst doc/index.rst controllers/main.py
sed -i "s/${REPO_NAME}/${REPO_NAME}/g" README.rst
sed -i "s/${ODOO_BRANCH}/${ODOO_BRANCH}/g" __manifest__.py
sed -i "s/${BRANCH}/${ODOO_BRANCH}/g" README.rst
sed -i "s/${TECHNICAL_NAME}/${TECHNICAL_NAME}/g" README.rst __manifest__.py
sed -i "s/${VERSION}/${ODOO_BRANCH}/g" README.rst __manifest__.py
sed -i "s/${ODOO_COMMIT_SHA_TO_BE_UPDATED}/${ODOO_REVISION}/g" README.rst

```

#

## 2.2 Guidelines

Source:

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

### 2.2.1 Comments

First of all, comments in the source are required if it's not obvious **why** are doing something.

Additionally, you can add comments about **what** are you doing, if it could be helpful.

## 2.3 Odoo Python

### 2.3.1 Python decorators

#### Original article

<http://odoo-new-api-guide-line.readthedocs.org/en/latest/decorator.html>

#### @api.one

**Warning:** the decorator is deprecated. Use @api.multi instead

api.one is meant to be used when method is called only on one record. It makes sure, that there are no multiple records when calling method with api.one decorator. Let say you got record partner = res.partner(1,). It is only one record and there is method for example (in res.partner):

```

@api.one
def get_name(self):
    return self.name #self here means one record

```

calling it like this works:

```
partner.get_name()
```

But if there would be more records, like:

```
partners = res.partner(1, 2,)
```

calling it, would raise Warning, telling you that you can only call it on one record.

### @api.multi

---

**Note:** Methods without decorators works the same way as @api.multi

---

something. For example:

```
@api.multi
def get_partner_names(self):
    names = []
    for rec in self:
        names.append(rec.name)
    return ', '.join(names)
```

### @api.model

And api.model is considered to be used when you need to do something with model itself and don't need to modify/check some exact model's record/records. For example there could be method that returns some meta info about model's structure or some helper methods, etc. Also in documentation it is said that this api is good to use when migrating from old api, because it "politely" converts code to new api. Also in my own experience, if you need method to return something, model decorator is good for it. api.one returns empty list, so it might lead to unexpected behavior when using api.one on method when it is supposed to return something.

## 2.3.2 Pure Python

### Compare two arrays

```
a = set(pos_config_obj.floor_ids.ids) b = set(rec.floor_ids.ids) diff = a.difference(b)
```

## 2.3.3 res.config.settings

Based on [https://github.com/odoo/odoo/blob/10.0/odoo/addons/base/res/res\\_config.py](https://github.com/odoo/odoo/blob/10.0/odoo/addons/base/res/res_config.py)

res.config.settings is a base configuration wizard for application settings. It provides support for setting default values, assigning groups to employee users, and installing modules. To make such a 'settings' wizard, define a model like:

```
class MyConfigWizard(models.TransientModel):
    _name = 'my.settings'
    _inherit = 'res.config.settings'
    default_foo = fields.type(..., default_model='my.model')
    group_bar = fields.Boolean(..., group='base.group_user', implied_group='my.group')
```

(continues on next page)

(continued from previous page)

```
module_baz = fields.Boolean(...)
other_field = fields.type(...)
```

The method `execute` (*Apply* button) provides some support based on a naming convention:

- For a field like `default_XXX`, `execute` sets the (global) default value of the field `XXX` in the model named by `default_model` to the field's value.
- For a boolean field like `group_XXX`, `execute` adds/removes ‘`implied_group`’ to/from the implied groups of ‘`group`’, depending on the field's value. By default ‘`group`’ is the group Employee. Groups are given by their `xml id`. The attribute ‘`group`’ may contain several `xml id`s, separated by commas.
- For a boolean field like `module_XXX`, `execute` triggers the immediate installation of the module named `XXX` if the field has value `True`.
- For the other fields, the method `execute` invokes all methods with a name that starts with `set_`; such methods can be defined to implement the effect of those fields.

The method `default_get` retrieves values that reflect the current status of the fields like `default_XXX`, `group_XXX` and `module_XXX`. It also invokes all methods with a name that starts with `get_default_`; such methods can be defined to provide current values for other fields.

## Example

```
from openerp import models, fields, api

PARAMS = [
    ("login", "apps_odoo_com.login"),
    ("password", "apps_odoo_com.password"),
]

class Settings(models.TransientModel):
    _name = 'apps_odoo_com.settings'
    _inherit = 'res.config.settings'

    login = fields.Char("Login")
    password = fields.Char("Password")

    @api.multi
    def set_params(self):
        self.ensure_one()

        for field_name, key_name in PARAMS:
            value = getattr(self, field_name, '').strip()
            self.env['ir.config_parameter'].set_param(key_name, value)

    def get_default_params(self, cr, uid, fields, context=None):
        res = {}
        for field_name, key_name in PARAMS:
            res[field_name] = self.env['ir.config_parameter'].get_param(key_name, '').
        strip()
        return res
```

## 2.3.4 Update settings on module install

To update settings from any `res.config.settings` do as follows:

### default\_XXX

TODO

### group\_XXX

Add **implied group(s)** to a **group** via `implied_ids` field:

```
<record model="res.groups" id="base.group_user">
    <field name="implied_ids" eval="[<br/>
        (4, ref('my.group'))<br/>
    ]"/>
</record>
```

### module\_XXX

Add XXX to the “depends” parameter in the `__manifest__.py` file.

### Other fields

Usually, other fields are saved to `ir.config_parameter`, so just *update ir.config\_parameter*, for example:

```
<function model="ir.config_parameter" name="set_param" eval="(pos_debt_notebook.debt_type, 'credit')"/>
```

## 2.3.5 Web controllers

### Send values to web page

If you need to transmit on rendering page some vars, you need to put that vars in dictionary and place it as second argument:

```
@http.route(['/shop/checkout'], type='http', auth="public", website=True)
def checkout(self, **post):
...
values['order'] = order
return request.website.render("website_sale.checkout", values)
```

## 2.3.6 One2One field in odoo

Odoo ORM doesn't support One2One fields, but you can do them manually. In the example below we make one2one relationship between models `fleet.vehicle` and `account.asset.asset`.

In short, you set normal Mane2one field (`vehicle_id` in the example) in a one model (doesn't really matter which of the models you choose) and corresponding One2many field (`asset_ids` in the example) in another model. Then we add virtual Many2one field (`asset_id` in the example) with attributes `compute` and `inverse`.

```
class Fleet(models.Model):
    _inherit = 'fleet.vehicle'
    ...
    asset_id = fields.Many2one('account.asset.asset', compute='compute_asset', inverse=
        ~'asset_inverse')
    asset_ids = fields.One2many('account.asset.asset', 'vehicle_id')

    @api.one
    @api.depends('asset_ids')
    def compute_asset(self):
        if len(self.asset_ids) > 0:
            self.asset_id = self.asset_ids[0]

    @api.one
    def asset_inverse(self):
        if len(self.asset_ids) > 0:
            # delete previous reference
            asset = self.env['account.asset.asset'].browse(self.asset_ids[0].id)
            asset.vehicle_id = False
            # set new reference
            self.asset_id.vehicle_id = self

class Asset(models.Model):
    _inherit = 'account.asset.asset'

    vehicle_id = fields.Many2one('fleet.vehicle', string='Vehicle')
```

TODO: replace `@api.one` to `@api.multi`

### 2.3.7 x2many values filling

To fill or manipulate one2many or many2many field with according values (records) you need to use special command as says below.

This format is a list of triplets executed sequentially, where each triplet is a command to execute on the set of records. Not all commands apply in all situations. Possible commands are:

- **(0, \_, values)** adds a new record created from the provided **value** dict.
- **(1, id, values)** updates an existing record of id **id** with the values in **values**. Can not be used in `~.create`.
- **(2, id, \_)** removes the record of id **id** from the set, then deletes it (from the database). Can not be used in `~.create`.
- **(3, id, \_)** removes the record of id **id** from the set, but does not delete it. Can not be used on `~openerp.fields.One2many`. Can not be used in `~.create`.
- **(4, id, \_)** adds an existing record of id **id** to the set. Can not be used on `~openerp.fields.One2many`.
- **(5, \_, \_)** removes all records from the set, equivalent to using the command **3** on every record explicitly. Can not be used on `~openerp.fields.One2many`. Can not be used in `~.create`.
- **(6, \_, ids)** replaces all existing records in the set by the **ids** list, equivalent to using the command **5** followed by a command **4** for each **id** in **ids**. Can not be used on `~openerp.fields.One2many`.

---

**Note:** Values marked as `_` in the list above are ignored and can be anything, generally `0` or `False`.

---

Taken from <https://github.com/odoo/odoo/blob/9.0/openerp/models.py>

## 2.3.8 Fields

Based on: <http://odoo-new-api-guide-line.readthedocs.io/en/latest/fields.html>

- *Field inheritance*
- *Field types*
  - *Boolean*
  - *Char*
  - *Text*
  - *HTML*
  - *Integer*
  - *Float*
  - *Date*
  - *Datetime*
  - *Binary*
  - *Selection*
  - *Reference*
  - *Many2one*
  - *One2many*
  - *Many2many*
- *Name Conflicts*
- *Fields Defaults*
- *Computed Fields*
- *Inverse*
- *Multi Fields*
- *Related Field*
- *Property Field*
- *WIP copyable option*
- *Special fields*
  - *active*

Now fields are class property:

```

from openerp import models, fields

class AModel(models.Model):

    _name = 'a_name'

    name = fields.Char(
        string="Name",                                     # Optional label of the field
        compute="_compute_name_custom",                   # Transform the fields in computed fields
        store=True,                                       # If computed it will store the result
        select=True,                                       # Force index on field
        readonly=True,                                     # Field will be readonly in views
        inverse="_write_name",                           # On update trigger
        required=True,                                     # Mandatory field
        translate=True,                                    # Translation enable
        help='blabla',                                     # Help tooltip text
        company_dependent=True,                          # Transform columns to ir.property
        search='_search_function'                        # Custom search function mainly used with
    )compute
)

# The string key is not mandatory
# by default it wil use the property name Capitalized

name = fields.Char()  # Valid definition

```

## Field inheritance

One of the new features of the API is to be able to change only one attribute of the field:

```
name = fields.Char(string='New Value')
```

## Field types

### Boolean

Boolean type field:

```
abool = fields.Boolean()
```

### Char

Store string with variable len.:

```
achar = fields.Char()
```

Specific options:

- size: data will be trimmed to specified size
- translate: field can be translated

## Text

Used to store long text.:

```
atext = fields.Text()
```

Specific options:

- translate: field can be translated

## HTML

Used to store HTML, provides an HTML widget.:

```
anhtml = fields.Html()
```

Specific options:

- translate: field can be translated

## Integer

Store integer value. No NULL value support. If value is not set it returns 0:

```
anint = fields.Integer()
```

## Float

Store float value. No NULL value support. If value is not set it returns 0.0 If digits option is set it will use numeric type:

```
afloat = fields.Float()  
afloat = fields.Float(digits=(32, 32))  
afloat = fields.Float(digits=lambda cr: (32, 32))
```

Specific options:

- digits: force use of numeric type on database. Parameter can be a tuple (int len, float len) or a callable that return a tuple and take a cursor as parameter

## Date

Store date. The field provides some helpers:

- `context_today` returns current day date string based on tz
- `today` returns current system date string
- `from_string` returns `datetime.date()` from string
- `to_string` returns date string from `datetime.date`

:

```
>>> from openerp import fields

>>> adate = fields.Date()
>>> fields.Date.today()
'2014-06-15'
>>> fields.Date.context_today(self)
'2014-06-15'
>>> fields.Date.context_today(self, timestamp=datetime.datetime.now())
'2014-06-15'
>>> fields.Date.from_string(fields.Date.today())
datetime.datetime(2014, 6, 15, 19, 32, 17)
>>> fields.Date.to_string(datetime.datetime.today())
'2014-06-15'
```

## Date

Store datetime. The field provide some helper:

- `context_timestamp` returns current day date string based on tz
- `now` returns current system date string
- `from_string` returns `datetime.date()` from string
- `to_string` returns date string from `datetime.date`

:

```
>>> fields.Datetime.context_timestamp(self, timestamp=datetime.datetime.now())
datetime.datetime(2014, 6, 15, 21, 26, 1, 248354, tzinfo=<DstTzInfo 'Europe/Brussels' _>
                CEST+2:00:00 DST>)
>>> fields.Datetime.now()
'2014-06-15 19:26:13'
>>> fields.Datetime.from_string(fields.Datetime.now())
datetime.datetime(2014, 6, 15, 19, 32, 17)
>>> fields.Datetime.to_string(datetime.datetime.now())
'2014-06-15 19:26:13'
```

## Binary

Store file encoded in base64 in bytea column:

```
abin = fields.Binary()
```

## Selection

Store text in database but propose a selection widget. It induces no selection constraint in database. Selection must be set as a list of tuples or a callable that returns a list of tuples:

```
aselection = fields.Selection([('a', 'A')])
aselection = fields.Selection(selection=[('a', 'A')])
aselection = fields.Selection(selection='a_function_name')
```

Specific options:

- selection: a list of tuple or a callable name that take recordset as input
- size: the option size=1 is mandatory when using indexes that are integers, not strings

When extending a model, if you want to add possible values to a selection field, you may use the *selection\_add* keyword argument:

```
class SomeModel(models.Model):  
    _inherits = 'some.model'  
    type = fields.Selection(selection_add=[('b', 'B'), ('c', 'C')])
```

## Reference

Store an arbitrary reference to a model and a row:

```
aref = fields.Reference([('model_name', 'String')])  
aref = fields.Reference(selection=[('model_name', 'String')])  
aref = fields.Reference(selection='a_function_name')
```

Specific options:

- selection: a list of tuple or a callable name that take recordset as input

## Many2one

Store a relation against a co-model:

```
arel_id = fields.Many2one('res.users')  
arel_id = fields.Many2one(comodel_name='res.users')  
an_other_rel_id = fields.Many2one(comodel_name='res.partner', delegate=True)
```

Specific options:

- comodel\_name: name of the opposite model
- delegate: set it to True to make fields of the target model accessible from the current model (corresponds to *\_inherits*)

## One2many

Store a relation against many rows of co-model:

```
arel_ids = fields.One2many('res.users', 'rel_id')  
arel_ids = fields.One2many(comodel_name='res.users', inverse_name='rel_id')
```

Specific options:

- comodel\_name: name of the opposite model
- inverse\_name: relational column of the opposite model

## Many2many

Store a relation against many2many rows of co-model:

```
arel_ids = fields.Many2many('res.users')
arel_ids = fields.Many2many(comodel_name='res.users',
                           relation='table_name',
                           column1='col_name',
                           column2='other_col_name')
```

Specific options:

- comodel\_name: name of the opposite model
- relation: relational table name
- columns1: relational table left column name (reference to record in current table)
- columns2: relational table right column name (reference to record in *comodel\_name* table)

In order to make two mutual many2many fields in different models use in them the same relation table and inverse columns:

```
_name = 'model1'
model2_ids = fields.Many2many(
    'model2', 'model2_ids_model1_ids_rel', 'model2_id', 'model1_id',
    _name = 'model2'
model1_ids = fields.Many2many(
    'model1', 'model1_ids_model2_ids_rel', 'model1_id', 'model2_id',
```

## Name Conflicts

---

**Note:** fields and method name can conflict.

---

When you call a record as a dict it will force to look on the columns.

## Fields Defaults

Default is now a keyword of a field:

You can attribute it a value or a function

```
name = fields.Char(default='A name')
# or
name = fields.Char(default=a_fun)

#...
def a_fun(self):
    return self.do_something()
```

Using a fun will force you to define function before fields definition.

Note. Default value cannot depend on values of other fields of a record, i.e. you cannot read other fields via `self` in the function.

## Computed Fields

There is no more direct creation of fields.function.

Instead you add a compute kwarg. The value is the name of the function as a string or a function. This allows to have fields definition atop of class:

```
class AModel(models.Model):
    _name = 'a_name'

    computed_total = fields.Float(compute='compute_total')

    def compute_total(self):
        ...
        self.computed_total = x
```

The function can be void. It should modify record property in order to be written to the cache:

```
self.name = new_value
```

Be aware that this assignation will trigger a write into the database. If you need to do bulk change or must be careful about performance, you should do classic call to write

To provide a search function on a non stored computed field you have to add a search kwarg on the field. The value is the name of the function as a string or a reference to a previously defined method. The function takes the second and third member of a domain tuple and returns a domain itself

```
def search_total(self, operator, operand):
    ...
    return domain # e.g. [('id', 'in', ids)]
```

## Inverse

The inverse key allows to trigger call of the decorated function when the field is written/"created"

## Multi Fields

To have one function that compute multiple values:

```
@api.multi
@api.depends('field.relation', 'an_otherfield.relation')
def _amount(self):
    for x in self:
        x.total = an_algo
        x.untaxed = an_algo
```

## Related Field

There is not anymore fields.related fields.

Instead you just set the name argument related to your model:

```
participant_nick = fields.Char(string='Nick name',
                                related='partner_id.name')
```

The `type` kwarg is not needed anymore.

Setting the `store` kwarg will automatically store the value in database. With new API the value of the related field will be automatically updated, sweet.

```
participant_nick = fields.Char(string='Nick name',
                                store=True,
                                related='partner_id.name')
```

---

**Note:** When updating any related field not all translations of related field are translated if field is stored!!

---

Chained related fields modification will trigger invalidation of the cache for all elements of the chain.

## Property Field

There is some use cases where value of the field must change depending of the current company.

To activate such behavior you can now use the `company_dependent` option.

A notable evolution in new API is that “property fields” are now searchable.

## WIP copyable option

There is a dev running that will prevent to redefine copy by simply setting a copy option on fields:

```
copy=False # !! WIP to prevent redefine copy
```

## Special fields

### active

TODO

See <https://github.com/odoo/odoo/blob/11.0/odoo/models.py#L3556-L3560>

## 2.3.9 Model constraints

Odoo provides two ways to set up automatically verified invariants: *Python constraints* <`openerp.api.constrains`> and *SQL constraints* <`openerp.models.Model._sql_constraints`>.

A Python constraint is defined as a method decorated with `~openerp.api.constrains`, and invoked on a recordset. The decorator specifies which fields are involved in the constraint, so that the constraint is automatically evaluated when one of them is modified. The method is expected to raise an exception if its invariant is not satisfied:

```
from openerp.exceptions import ValidationError

@api.constrains('age')
def _check_something(self):
    for record in self:
        if record.age > 20:
            raise ValidationError("Your record is too old: %s" % record.age)
    # all records passed the test, don't return anything
```

SQL constraints are defined through the model attribute `~openerp.models.Model._sql_constraints`. The latter is assigned to a list of triples of strings (`name`, `sql_definition`, `message`), where `name` is a valid SQL constraint name, `sql_definition` is a `table_constraint_expression`, and `message` is the error message.

### 2.3.10 Reports models via PostgreSQL views

Postgres View is a kind of table, which is not physically materialized. Instead, the query is run every time the view is referenced in a query.

To create Postgres View in odoo do as follows:

- create new model
- all fields must have the flag `readonly=True`.
- specify the parameter `_auto=False` to the odoo model, so no table corresponding to the fields is created automatically.
- add a method `init(self, cr)` that creates a PostgreSQL View matching the fields declared in the model.
  - `id` field has to be specified in `SELECT` part. See example below
- add views for the model in a usual way

Example:

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

class ReportEventRegistrationQuestions(models.Model):
    _name = "event.question.report"
    _auto = False

    attendee_id = fields.Many2one(comodel_name='event.registration', string=
        'Registration')
    question_id = fields.Many2one(comodel_name='event.question', string='Question')
    answer_id = fields.Many2one(comodel_name='event.answer', string='Answer')
    event_id = fields.Many2one(comodel_name='event.event', string='Event')

@api.model_cr
def init(self):
    """ Event Question main report """
    tools.drop_view_if_exists(self._cr, 'event_question_report')
    self._cr.execute(""" CREATE VIEW event_question_report AS (
        SELECT
            att_answer.id as id,
            att_answer.event_registration_id as attendee_id,
            answer.question_id as question_id,
            answer.id as answer_id,
            question.event_id as event_id
        FROM
            event_registration_answer as att_answer
        LEFT JOIN
            event_answer as answer ON answer.id = att_answer.event_answer_id
        LEFT JOIN
            event_question as question ON question.id = answer.question_id
        GROUP BY
            attendee_id,
            event_id,
            question_id,
            answer_id
    )""")
```

(continues on next page)

(continued from previous page)

```
question_id,
answer_id,
att_answer.id
) """)
```

### 2.3.11 External dependencies in odoo

#### What

External dependencies are python packages or any binaries, that have to be installed to make module work.

#### How

In python files where you use external dependencies you will need to add `try-except` with a debug log.

```
import

try:
    import external_dependency_python_N
    import external_dependency_python_M
except ImportError as err:
    _logger.debug(err)

# for binary dependencies:
try:
    import external_dependency_python_N
    import external_dependency_python_M
except IOError as err:
    _logger.debug(err)
```

This rule doesn't apply to the test files since these files are loaded only when running tests and in such a case your module and their external dependencies are installed.

Also, you need to add external dependencies to *manifest*.

#### Why

Odoo loads python files of a module whenever following conditions are satisfied:

- the module has static folder (e.g. for `icon.png`)
- the module marked as installable in *manifest*, i.e. the module *can* be installed

One can see, that odoo loads python files even if module is not installed (and even not intended to be installed). But modules usually are added to addons-path as a part of some repository (e.g. `pos-addons`). This is why importing external dependencies without `try-except` leads to problems on adding repository to *addons-path*.

## 2.4 XML

### 2.4.1 Create record of model

Create new record:

```
<openerp>
    <data>
        <record id="demo_multi_session" model="pos.multi_session">
            <field name="name">multi session demo</field>
        </record>
    </data>
</openerp>
```

If model exist it will be modified. Record creating in module it declared. To change model created in another module add mule name before id:

```
<openerp>
    <data>
        <record id="point_of_sale.pos_config_main" model="pos.config">
            <field name="multi_session_id" ref="demo_multi_session"/>
        </record>
    </data>
</openerp>
```

## 2.4.2 Xpath

### Add some attributes to node

Code:

```
<xpath expr="//some/xpath" position="attributes">
    <attribute name="some_field">
</xpath>
```

Qweb expression:

```
<attribute name="t-att-another_field">website.get_another_field_value()</attribute>
```

After rendering it becomes regular attribute:

```
<.... another_field="value" ...>
```

### Important

Inside of

```
<xpath expr="//some/xpath" position="attributes">
    ...
</xpath>
```

you can put **only** `<attribute name=` and nothing more.

### To test xpath

Code:

```
#Odoo tip - XPath playground:  
$ sudo apt-get install libxml-xpath-perl  
$ xpath -e "//record[@id=',,,']" -e "//field[@name='...']" *.xml
```

## 2.4.3 Basic stuff

### Call method of some model and put result in variable

Code:

```
<t t-set="order" t-value="website.sale_get_order()"/>
```

Here *website* means you use website=True in controller. TODO my be wrong.

### Get value of some setting ir.config\_parameter and put it in variable

Code:

```
<t t-set="foobar" t-value="website.env['ir.config_parameter'].get_param('my_module.  
foobar')"/>
```

### Show value of variable

Code:

```
<p><t t-esc="foobar"/></p>
```

### Use variable in condition

Code:

```
<label t-if="foobar">  
    <p>foobar is true</p>  
</label>
```

### Get variable transmitted by render() in XML template

Code:

```
t-att-value="my_var"
```

*my\_var* is element of ‘values’ dictionary (second argument of render()).

## 2.4.4 Inherit

### Collisions and priority

If two or more xml templates inherit same parent template they can have same priorities. It may produce conflicts and unexpected behavior. What you need is just set priority explicitly in your template:

```
<template id="..." inherit_id="..." priority="8" ...>
    <xpath expr="..." position="...">
        ...
    </xpath>
</template>

<!-- or -->

<record id="..." model="ir.ui.view">
    ...
    <field name="inherit_id" ref="..."/>
    <field name="priority" eval="8" />
    <field name="arch" type="xml">
        <xpath expr="..." position="...">
        </xpath>
    </field>
</record>
```

Less priority means prior execution.

Default priority is 16.

## 2.4.5 Group\_id in views

groups\_id in views (mostly in views that inherits other views) allows to specify for which group make those inheritance.

For example, if you need to add a button that will be available for managers only, make the following:

```
<record id="fleet_vehicle_log_services_form_inherit_bos" model="ir.ui.view">
<field name="name">fleet.vehicle.log.services.form.inherit.bos</field>
<field name="model">fleet.vehicle.log.services</field>
<field name="inherit_id" ref="fleet_vehicle_log_services_form_inherit_buttons"/>
<field name="groups_id" eval="[(4, ref('fleet_booking.group_branch_officer'), 0)]"/>
<field name="arch" type="xml">
    <data>
        <xpath expr="//button[@name='confirm']" position="attributes">
            <attribute name='attrs'>{'invisible': [('cost_subtype_in_branch', '=', _False)]}</attribute>
        </xpath>
    </data>
</field>
</record>
```

If groups\_id is omitted, then the update is applied for all users.

## 2.5 HTML

### 2.5.1 Active elements

#### Link-button that calls controller

Code:

```
<form action="/shop/checkout" name="myform" method="post">
    <a class="btn btn-primary a-submit">My button</a>
</form>
```

Here action="/shop/checkout" sets controller address. Class a-submit usually means do what in 'action' of form.

## Submit with button

Code:

```
<form action="/my_page" name="myform" method="post">
    <button type="submit" class="btn btn-default">My button</button>
</form>
```

Wherein in controller in \*\*post will be available some values from source form, those like <input>.

## 2.6 CSS

### 2.6.1 CSS tips and tricks

#### Add your css on template

Code:

```
<template id="my_module_frontend" name="my_module assets" inherit_id="website_sale.
↪assets_frontend">
    <xpath expr="//link[@rel='stylesheet']" position="after">
        <link rel="stylesheet" href="/my_module/static/src/css/main.css"/>
    </xpath>
</template>
```

*website\_sale.assets\_frontend* is what you inherits.

#### Hide fields

Hide all children (that have attribute bill='1') of oe\_website\_sale class owner (that have attribute bill\_enabled='0'):

```
.oe_website_sale[bill_enabled='0'] [bill='1']{
    display:none;
}
```

## 2.7 YAML

### 2.7.1 Pure YAML

TODO

## 2.7.2 YAML in odoo

TODO

# 2.8 Javascript

## 2.8.1 Inheritance

TODO

## 2.8.2 core.bus

core.bus (web.bus in 8.0) is used handle js events between modules.

### Usage

```
// 8.0
var bus = openerp.web.bus;

// 9.0+
var core = require('web.core');
var bus = core.bus;

// bind event handler
bus.on('barcode_scanned', this, function (barcode) {
    //...
})

// trigger event
bus.trigger('barcode_scanned', barcode);
```

## 2.8.3 Remote Procedure Call (RPC)

### Call method

```
/**
 * Call a method (over RPC) on the bound OpenERP model.
 *
 * @param {String} method name of the method to call
 * @param {Array} [args] positional arguments
 * @param {Object} [kwargs] keyword arguments
 * @param {Object} [options] additional options for the rpc() method
 * @returns {jQuery.Deferred<>} call result
 */
call: function (method, args, kwargs, options) {
    args = args || [];
    kwargs = kwargs || {};
    if (!_.isArray(args)) {
        // call(method, kwargs)
        kwargs = args;
    }
}
```

(continues on next page)

(continued from previous page)

```

        args = [];
    }
var call_kw = '/web/dataset/call_kw/' + this.name + '/' + method;
return session.rpc(call_kw, {
    model: this.name,
    method: method,
    args: args,
    kwargs: kwargs
}, options);
},

```

## How to call wizard method from js

```

var compose_model = new Model('mail.compose.message');
return compose_model.call('create', [msg, {default_parent_id: options.parent_id}])
.then(function(id){
    return compose_model.call('send_mail_action', [id, {}]);
});

```

## 2.9 Frontend

### 2.9.1 Web page

#### Common

Open a new project:

```
./odoo.py scaffold newpage addons
```

Add website as a dependency to newpage:

```
'depends': 'website'
```

then add the website=True flag on the controller, this sets up a few new variables on the request object and allows using the website layout in our template.

#### Creating pages

##### 1 way

Write the following code in controllers.py:

```

from openerp import http
classNewPage(http.Controller):
    @http.route('/new-page/', auth='public', website=True)
    def index(self, **kw):
        return http.request.render('newpage.index')

```

The new web page will appear by adding - /new-page/ http.request.render('newpage.index') – downloading a tamplate for a new page

A pattern templates.xml

```
<openerp>
    <data>
        <templateid="index">
            <t t-call="website.layout">
                <t t-set="title">New page</t>
                <div class="oe_structure">
                    <div class="container">
                        <h1>My first web page</h1>
                        <p>Hello, world!</p>
                    </div>
                </div>
            </t>
        </template>
    </data>
</openerp>
```

`website.layout` means that the elements of pattern website are used.

After restarting the server while updating the module (in order to update the manifest and template) access <http://localhost:8069/new-page/>. You will see a new page with a title '*My first web page*' and with text '*Hello, world!*'

## 2 way

Write in pattern the following:

```
<template name="Services page" id="website.services" page="True">
    <t t-call="website.layout">
        <div id="wrap">
            <div class="container">
                <h1>Our Services</h1>
                <ul class="services">
                    <li>Cloud Hosting</li>
                    <li>Support</li>
                    <li>Unlimited space</li>
                </ul>
            </div>
        </div>
    </t>
</template>
```

`page="True"` creates a page as follows below: <http://localhost:8069/page/services/>

If add in `view.xml`:

```
<record id="services_page_link" model="website.menu">
    <field name="name">Services</field>
    <field name="url">/page/services</field>
    <field name="parent_id" ref="website.main_menu" />
    <field name="sequence" type="int">99</field>
</record>
```

This code will add a link to the main menu.

## 2.10 Point of Sale (POS)

### 2.10.1 New POS Module

## Adding js file to POS

Adding **javascript** file opens a new set of possibilities in Odoo.

Let take the example of the POS Debt & Credit notebook module:

```
<template id="assets" inherit_id="point_of_sale.assets">
    <xpath expr=". position="inside">
        <script type="text/javascript" src="/pos_debt_notebook/static/src/js/pos.js" />
        <link rel="stylesheet" href="/pos_debt_notebook/static/src/css/pos.css" id="pos_
        debt_notebook-stylesheet" />
    </xpath>
</template>
```

## odoo.define function

Official doc about the topic is [here](#):

*Javascript Module* in odoo is some piece of code declared via `odoo.define ('js_module_name', ...)` and can be used in other modules via `require ('js_module_name')`.

### Example:

```
odoo.define('js_module_name', function (require) {
    "use strict";
    var A = require('js_module_name_A');
    var B = require('js_module_name_B');
    require('js_module_name_C');

    // some code
    return something;
});
```

**Warning:** You cannot rename variable `require`.

---

**Note:** Single file may have several *JS modules*, though it's recommended to put them to different files

---



---

**Note:** You can use any string as a module name, but recommended way is `<OODOO_MODULE>. <JS_MODULE>`, e.g. `point_of_sale.popups`

---

## Return value

A js-module may return value. That value can be used in another js-modules (of the same odoo-module or others).

For example:

```
odoo.define('point_of_sale.gui', function (require) {
    "use strict";
    return {
        Gui: Gui,
```

(continues on next page)

(continued from previous page)

```
define_screen: define_screen,
define_popup: define_popup,
};

});
```

Then, we can use `define_screen` as following:

```
odoo.define('point_of_sale.screens', function (require) {
  "use strict";
  var gui = require('point_of_sale.gui');
  //...
  gui.define_screen({
    name: 'scale',
    widget: ScaleScreenWidget
  });
  // ...
  return ....
})
```

---

**Note:** If you don't want to use value returned by another js-module, you still might need to *import js-module* (via `require(...)`) to be sure that that module is loaded before executing your module.

---

## Asynchronous modules

It can happen that a module needs to perform some work before it is ready. For example, it could do a rpc to load some data. In that case, the module can simply return a deferred (promise). In that case, the module system will simply wait for the deferred to complete before registering the module.

```
odoo.define('module.Something', function (require) {
  "use strict";
  var ajax = require('web.ajax');
  return ajax.rpc(...).then(function (result) {
    // some code here
    return something;
  });
})
```

## Inheritance

POS has two types of classes: Models, Widget. Extending those classes are slightly different.

---

**Note:** Not all classes have an easy way to get them to inherit. Some tricks are available [here](#).

---

### Model

Model classes work with data only and don't work with UI directly.

To extend that kind of class, you need to use `extend` method. It creates a copy of class with redefined method. Normally, you need to override original class with updated one. Also, to call original method, put original class to a variable.

Here is an example:

```
odoo.define('pos_debt_notebook.pos', function (require) {
    "use strict";

    var models = require('point_of_sale.models');

    // save original class
    var _super_posmodel = models.PosModel.prototype;
    // override original class with extended one
    models.PosModel = models.PosModel.extend({
        initialize: function (session, attributes) {
            var self = this;
            // some new code in this method
            models.load_fields('product.product', ['credit_product']);
            // call original method via "apply"
            _super_posmodel.initialize.apply(this, arguments);
        },
    })
})
```

## Widget

Widget classes work with UI.

Widget extend is much easier than Model extending: just use `include` and `_super`.

Here is an example:

```
odoo.define('pos_debt_notebook.pos', function (require) {
    "use strict";
    var screens = require('point_of_sale.screens');

    // "include" updates original method
    screens.PaymentScreenWidget.include({
        init: function (parent, options) {
            // call super in a easy way
            this._super(parent, options);
            // add some new code
            this.pos.on('updateDebtHistory', function (partner_ids) {
                this.update_debt_history(partner_ids);
            }, this);
        },
    })
})
```

## 2.10.2 UI extending

### Action Buttons

Action Buttons are following buttons:

- Note

- Transfer
- Guests
- Bill
- Split
- Order
- Discount
- etc.

that are located above **Numpad**.

- These buttons only show up after installing **pos\_discount** module (Discount button, which allows defining the size of discount for the order) and **pos\_restaurant** module (Split, Guests buttons etc.)
- You can create your own buttons assigning them to actions (for example **open\_popup**, **screen** and etc).
- To create a Button you need to inherit **ActionButtonWidget** Class, select **define\_action\_button** and choose the necessary action after the pressing corresponding button.

Consider, for example, the way of creating a simple Button, which opens popup-window.

```
odoo.define('pos_popup_button', function (require) {
    'use strict';
    /*
    In order to use ActionButtonWidget, which specified in Screens
    please start with downloading the screens widget
    */

    var screens = require('point_of_sale.screens');

    //declare a new variable and inherit ActionButtonWidget

    var PopupButton = screensActionButtonWidget.extend({
        /*
        Thus PopupButton contains all methods from ActionButtonWidget.
        Now we need to define Template for our button,
        where the type of button you can find in Qweb (see below)
        */
        template: 'PopupButton',
        /*
        We also need to choose the Action,
        which will be executed after we click the button.
        For this purpose we define button_click method, where
        where name - Button name; widget - Button object;
        condition - Condition, which calls the button to show up
        (in our case, setting on show_popup_button option in POS config).
        */
        button_click: function () {
            this.gui.show_popup('confirm', {
                'title': 'Popup',
                'body': 'Opening popup after clicking on the button',
            });
        }
    });

});
```

(continues on next page)

(continued from previous page)

```
screens.define_action_button({
    'name': 'popup_button',
    'widget': PopupButton,
    'condition': function () {
        return this.pos.config.popup_button;
    },
});
return PopupButton;
});
```

The definition of the template in Qweb:

```
<t t-name="PopupButton">
<div class="control-button">
    <i class="fa fa-list-alt" /> Popup Button
</div>
</t>
```

For a concrete example check the **POS Orders History** module , where you can see that a button with the label *Orders History* is added.

## Creation of Custom Pop-Ups

Use the Custom Pop-Ups to provide information or to prompt Users to do something in POS. You can define the appearance of a pop-up.

**Let take the example of the creation a pop-up of QR Code Scanning in POS module** , where we needed to create a pop-up to show the video from camera to scan QR codes.

First, attach necessary requirements:

```
var PopupWidget = require('point_of_sale.popups');
```

Then, create a new instance, the new default pop-up extension:

```
var QrScanPopupWidget = PopupWidget.extend({
// It requires the template attribute with a Qweb template name for showing pop-up
template: 'QrScanPopupWidget',
```

To make the pop-up be reachable with regular methods after the `QrScanPopupWidget` declaration do the following:

```
gui.define_popup({name: 'qr_scan', widget: QrScanPopupWidget});
```

so it can be called with the next code:

```
this.gui.show_popup('qr_scan', {
    'title': 'QR Scanning',
    'value': false,
});
```

## Custom Screens

List of **partners**, **payment's screen**, and **floor screen** are examples of screens.

We will consider *an example of creating the User interface*.

In order to create a new custom screen we plug screens and gui:

```
var screens = require('point_of_sale.screens');
var gui = require('point_of_sale.gui');
```

Then we declare a new variable and inherit ScreenWidget:

```
var CustomScreenWidget = screens.ScreenWidget.extend({});
```

Now CustomScreenWidget consist of all methods from ScreenWidget. Then we need to define a template, where the structure of the screen is described using Qweb:

```
template: 'CustomScreenWidget'
```

In Qweb we define a template as follows:

```
<t t-name="CustomScreenWidget">
  <div class="custom-screen screen">
    <div class="screen-content">
      <section class="top-content">
        <span class="button back">
          <i class="fa fa-angle-double-left" />
          Cancel
        </span>
        <span class="button next oe_hidden highlight">
          Apply
          <i class="fa fa-angle-double-right" />
        </span>
      </section>
      <section class="full-content">
        <div class="window">
          <section class="subwindow collapsed">
            <div class="subwindow-container collapsed">
              <div class="subwindow-container-fix custom-details-contents" />
            </div>
          </section>
        </div>
      </section>
    </div>
  </t>
```

Define styles in css file, which you need for the screen.

This Qweb will be rendered every time when the method `renderElement` runs (prior to the downloading POS all screens are drawn and hidden already). This method can be redefine and, for example, used for actions of back and next buttons:

```
renderElement: function () {
  this._super();

  this$('.back').click(function () {
    self.gui.back();
  });

  this$('.next').click(function () {
    // some actions
  });
}
```

(continues on next page)

(continued from previous page)

```
    });
},
```

All screens are hidden by default (except those, which are called after POS downloading). In order to open Custom Screens you need to define it inside screens' list:

```
gui.define_screen({name: 'custom_screen', widget: CustomScreenWidget});
```

In order to open Custom Screen you need to call the next function (for example after click to the Action button):

```
this.gui.show_screen('custom_screen');
```

where this is a pointer to PosModel.

For a concrete example check the **POS Orders History** module , where orders\_history\_screen is defined.

### 2.10.3 Receipts & Printers

#### Custom Receipt

There are two types of receipts in POS:

1. PosTicket - displays on screen after payment;
2. XmlReceipt - prints in PosBox after pressing the button Print receipt on the Payment screen if PosBox connected with the ESC POC printer.

These two receipts are implemented on Qweb and generated after purchase order. If PosTicket allows any design in Qweb, then for XmlReceipt you can use only strictly defined tags, which supported by the ESC POC printer.

Using t-extend mechanism, which takes the template's name to be modified as a parameter, you can extend existing Qweb templates for receipts. After that, a modification with any number t-jquery sub-directives can be performed.

t-jquery directives use the CSS selector. This selector is used in the extended template for choosing context nodes, for which t-operation can be applied. If you want to add, for example, another title for Product in XmlReceipt, you need to create Qweb with the following content:

```
<t t-extend="XmlReceipt">
  <t t-jquery="t[t-if='simple'] line" t-operation="after">
    <t t-set="second_product_name" t-value="line.second_product_name"/>
      <t t-if="pos.config.show_second_product_name_in_receipt and second_product_name
  </t>
    <line>
      <left>(<t t-esc='second_product_name' />)</left>
    </line>
  </t>
</t>
</t>
```

For the same action for PosTicket:

```
<t t-extend="PosTicket">
  <t t-jquery=".receipt-orderlines tr[t-foreach='orderlines']" t-operation="append">
    <t t-set="second_product_name" t-value="orderline.get_product().second_product_name
  </t>
```

(continues on next page)

(continued from previous page)

```
</t>
</t>
```

One of the difficult but at the same time flexible method of creating Customer receipt is to download the field from the Server, where Qweb template is described in the text form. After downloading and before printing this text template need to be converted into XML format and produced data based on this template.

```
template_receipt_qweb = fields.Text(string="Custom Receipt Qweb")
```

In POS you need to convert this text format into XML and generate a receipt using this template:

```
convert_to_xml: function (template) {
    var parser = new DOMParser();
    var xmlDoc = parser.parseFromString(template, "text/xml");
    return xmlDoc.documentElement;
},
```

Usage of this template instead of standards ones requires to generate received XML, in order to do this you need to connect Qweb:

```
var core = require('web.core');
var Qweb = core.qweb;
```

To define a custom function, which will generate a user's Qweb as follows:

```
custom_qweb_render: function (template, options) {
    var template_name = $(template).attr('t-name');
    Qweb.templates[template_name] = template;
    return Qweb._render(template_name, options);
},
```

This function needs to be called every time when the receipt is generated.

## 2.10.4 Extra Order Data

### Loading data to POS

By default POS uploads next models:

```
res.users, res.company, decimal.precision, uom.uom, res.partner, res.country,
account.tax, pos.session, pos.config, res.users, stock.location, product.pricelist,
product.pricelist.item, product.category, res.currency, pos.category, product.
product, account.bank.statement, account.journal, account.account.fiscal.position, account.
fiscal.position.tax.
```

If we've added a new field in the backend and want them to be presented in the POS we can use **load\_fields** method inside the PosModel **initialize** function.

In the next example taken from POS Debt & Credit notebook module we add some new fields to the account.journal model:

```
var models = require('point_of_sale.models');
models.load_fields('account.journal', ['debt', 'debt_limit', 'credits_via_discount',
    'pos_cash_out',
    'category_ids', 'credits_autopay']);
```

In order to upload a new model into POS we use `load_models(models, options)`. Description's taken from `odoo`.

Loads openerp models at the point of sale startup.

`load_models` take an array of model loader declarations.

The models will be loaded in the array order. If no openerp model name is provided, no server data will be loaded, but the system can be used to preprocess data before load.

Loader arguments can be functions that return a dynamic value. The function takes the `PosModel` as the first argument and a temporary object that is shared by all models, and can be used to store transient information between model loads.

There is no dependency management. The models must be loaded in the right order. Newly added models are loaded at the end but the `after` / `before` options can be used to load directly before / after another model.

```
models: [
    model: [string] the name of the openerp model to load.
    label: [string] The label displayed during load.
    fields: [[string]|function] the list of fields to be loaded.
        Empty Array / Null loads all fields.
    order: [[string]|function] the models will be ordered by the provided fields
    domain: [domain|function] the domain that determines what
        models need to be loaded. Null loads everything
    ids: [[id]|function] the id list of the models that must
        be loaded. Overrides domain.
    context: [Dict|function] the openerp context for the model read
    condition: [function] do not load the models if it evaluates to
        false.
    loaded: [function(self,model)] this function is called once the
        models have been loaded, with the data as second argument
        if the function returns a deferred, the next model will
        wait until it resolves before loading.
]

options:
    before: [string] The model will be loaded before the named models
        (applies to both model name and label)
    after: [string] The model will be loaded after the (last loaded)
        named model. (applies to both model name and label)
```

Example below uploads all records meet the domain `account.invoice` model.

The **loaded** function is a handler for uploaded data.

Here you can proceed and save this example which is taken from Pay Sale Orders & Invoices over POS module:

```
var models = require('point_of_sale.models');
models.load_models({
    model: 'account.invoice',
    fields: ['name', 'partner_id', 'date_invoice', 'number', 'date_due', 'origin',
    ↵'user_id', 'residual', 'state', 'amount_untaxed', 'amount_tax'],
    domain: [['state', '=', 'open'], ['type', '=', 'out_invoice']],
    loaded: function (self, invoices) {
        var invoices_ids = _.pluck(invoices, 'id');
        self.prepare_invoices_data(invoices);
        self.invoices = invoices;
        self.db.add_invoices(invoices);
```

(continues on next page)

(continued from previous page)

```
        self.get_invoice_lines(invoices_ids);
    }
});
```

## Custom order data in browser storage

Before the payment orders in POS are kept in browser storage. Thereby if we again open POS module (should not be confused with the closing of the session) the system automatically retrieves data from the storage.

If your model adds data (of the field), then you need to make additional data processing in order to save this data among reopenings.

Because of the browser storage (*localStorage*) allows saving data only with the type String POS converts the Order object to the String and inversely.

For this purpose following methods are used:

- `init_from_JSON`: function reads parameters of the order from the *json-String* and saves to the current object (see realization of the Orderline [here](#) and for the Model [there](#))
- `export_as_JSON`: function converts the current object to *json-String* (see realization of the Orderline [here](#) and for the Model [there](#))

When order is updated `export_as_JSON` is called and data are saved to browser storage.

Now, if you close page and reopen, then at some point `init_from_JSON` is called to restore order from *json string*.

Let's take the example:

### POS Advanced Order Notes module

This [module](#) allows adding notes to the entire order, to use already predefined notes and to speed up the process of creating orders by specifying products also via notes, which can be automatically applied further.

```
export_as_JSON: function () {
    var data = _super_order.export_as_JSON.apply(this, arguments);
    data.note = this.note;
    return data;
},
init_from_JSON: function (json) {
    this.note = json.note;
    _super_order.init_from_JSON.call(this, json);
},
```

When you add the note to your Order, the trigger which calls `export_as_JSON` launches to convert data of current order into string (including notes) and save it in browser storage.

While loading POS in order to get saved notes from the browser storage, you need to extend `init_from_JSON` function.

Without this code, your module will work, but if you reload POS, your notes will not be presented (because they are not saved).

## Sending POS Orders to Server

This article describes the process of sending POS Orders to odoo server and demonstrates possible usage of extending it.

The general process is as follows:

Client side:

- `export_as_JSON`: converts *order data* to send to the server
- then *order* is saved to browser cache
- then POS tries to send data to server

Backend side:

- `create_from_ui`: data come to POS (see [here](#))
- `_process_order`: process order json (created records in database, etc. see [here](#))
- `_order_fields`: prepare dictionary for create method (see [how](#))

So, in order to pass additional information and handle it on server we need:

- extend `export_as_JSON` in client side
- extend `_process_order` in server side

Let's check it on example:

### Saving removed products of POS order module

The module allows to add a reason on canceling order or orderline in POS.

In order to do it we:

- extend `export_as_JSON` in client side (see [here](#))

```
export_as_JSON: function() {
    var data = _super_order.export_as_JSON.apply(this, arguments);
    /* canceled_lines is used only on the client side
     * to cache those data in order to prevent misbehavior
     * in case the page was refreshed
    */
    data.canceled_lines = this.canceled_lines || [];
    // update data to be sent to the server
    data.reason = this.reason;
    data.is_cancelled = this.is_cancelled;
    return data;
},
```

- extend `_process_order` in server side (see [here](#))

```
@api.model
def _process_order(self, pos_order):
    order = super(PosOrder, self).process_order(pos_order)
    if 'is_cancelled' in pos_order and pos_order['is_cancelled'] is True:
        if pos_order['reason']:
            order.cancellation_reason = pos_order['reason'].encode('utf-8')
            order.is_cancelled = True
    return order
```

## 2.10.5 Instant synchronization

### POS Longpolling

It is a custom odoo [module](#) made by [IT-Projects LLC](#), which allows sending instant updates to the POS interfaces from backend.

It provides following methods in *Backend side*:

- `self.env['pos.config'].send_to_all_poses(channel_name, data)`: broadcasts messages to all opened POSEs (see [example](#))
- `pos_set._send_to_channel(channel_name, data)`: broadcasts message to the POSEs in `pos_set` (see [example](#))
- `_send_to_channel_by_id(self, dbname, pos_id, channel_name)`: sends message to exact POS `pos_id`, uses data base name `dbname`, `channel_name`, `message='PONG'` (see [example](#))

---

**Note:** POS will get notification only if it's subscribed to the specified `channel_name`.

---

For *Client side* the methods are:

- `add_bus(key, sync_server)`: allows to create additional Bus to sync data with another Sync Server (see [example](#) in `pos_multi_session` - it gets data from local server to speed up synchronization)

---

**Note:** You don't need to use `add_bus` if you connect with your regular odoo server.

---

- `add_channel_callback: function(channel_name, callback, thisArg)`: subscribes to specific channel (see [example](#))

Let's check example of usage taking as a basis `Sync Partners` in POS module:

### Sync Partners in POS module

This [module](#) on each partner update (in Backend) notifies POSEs to update partner data.

Here you can see how it uses `pos_longpolling`:

## BACKEND

- On partner update `send_field_updates` [method](#) is called:

```
@api.model
def send_field_updates(self, partner_ids, action=''):
    channel_name = "pos_partner_sync"
    data = {'message': 'update_partner_fields', 'action': action, 'partner_ids': _.
    ↪partner_ids}
    self.env['pos.config'].send_to_all_poses(channel_name, data)
```

- It uses `send_to_all_poses` method.

## CLIENT

- On POS starting it's subscribed to channel pos\_partner\_sync.

```
initialize: function () {
    PosModelSuper.prototype.initialize.apply(this, arguments);
    var self = this;
    this.ready.then(function () {
        self.bus.add_channel_callback("pos_partner_sync", self.on_barcode_updates, self);
    });
},
```

- On notification on\_barcode\_updates is called, which reloads partner data:

```
on_barcode_updates: function(data) {
    var self = this;
    if (data.message === 'update_partner_fields') {
        var def = new $.Deferred();
        if (data.action && data.action === 'unlink') {
            // partner is deleted. Make necessary updates in UI
            this.remove_unlinked_partners(data.partner_ids);
            def.resolve();
        } else {
            // reload partner data
            def = self.load_new_partners(data.partner_ids);
        }
        return def.done(function() {
            var opened_client_list_screen = self.gui.get_current_screen() === 'clientlist' &
            ↪ self.gui.screen_instances.clientlist;
            if (opened_client_list_screen) {
                // rerender partner list
                opened_client_list_screen.update_client_list_screen(data.partner_ids);
            }
        });
    }
},
```

## Multi-session Support

pos\_multi\_session is a module, which allows synchronizing data in POSes within one multi\_session.

In order to synchronize new user data Order or Orderline models of one POS with others, you no need to add a new module pos\_multi\_session into depends on your module, you need to extend such methods as export\_as\_JSON, init\_from\_JSON and add the method apply\_ms\_data, which is used for compatibility with .

Consider the **Example of synchronization for the Order model**.

Let us have some data for the order and we need to synchronize it with all POSes, which use the same multi-session:

```
apply_ms_data: function (data) {
    /*
    It is necessary to check the presence of the super method
    in order to be able to inherit the apply_ms_data
    without calling require('pos_multi_session')
    and without adding pos_multi_session in dependencies in the manifest.
```

(continues on next page)

(continued from previous page)

```
At the time of loading, the super method may not exist. So, if the js file is loaded
first among all inherited, then there is no super method and it is not called.
If the file is not the first, then the super method is already created by other
→modules,
and we call super method.
*/
if (_super_order.apply_ms_data) {
    _super_order.apply_ms_data.apply(this, arguments);
}
this.first_new_variable = data.first_new_variable;
this.second_new_variable = data.second_new_variable;
// etc ...

/*
Call renderElement directly or trigger corresponding
event if you need to rerender something after updating */
},
export_as_JSON: function () {
// export new data as JSON
var data = _super_order.export_as_JSON.apply(this, arguments);
data.first_new_variable = this.first_new_variable;
data.second_new_variable = this.second_new_variable;
return data;
},
init_from_JSON: function (json) {
// import new data from JSON
this.first_new_variable = json.first_new_variable;
this.second_new_variable = json.second_new_variable;
return _super_order.init_from_JSON.call(this, json);
}
```

## 2.10.6 Advanced POS Development

### Dom Cache

Dom Cache is used to save rendered elements to speed POS up.

To add something to Dom Cache you need to do something like this:

```
this.cache = new screens.DomCache();
this.cache.cache_node(key, value);
```

To restore rendered element from cache do something like this

```
this.cache = new screens.DomCache();
var cache = this.cache.get_node(key);
```

Here is complete example from Point of Sale module:

The purpose of this code is the optimization of the elements rendering in POS. Each new POS loading use data from DomCache - thereby save time for the rendering of new elements.

Let's take the example:

## POS Order History module

In this in this module DomCache is used when the orders' list renders.

After the first loading POS elements of orders, which have been rendered (*HTML code*), are saved in Cache.

After reloading POS the existence of saved elements in Cache are checked and this data is used when orders are rendered.

```
init: function(parent, options) {
  this._super(parent, options);
  //object of DomCache, which we will use in order to address the methods of this_
  //object
  this.orders_history_cache = new screens.DomCache();
},
render_list: function(orders) {
  var contents = this.$el[0].querySelector('.order-list-contents');
  contents.innerHTML = "";
  for (var i = 0, len = Math.min(orders.length, 1000); i < len; i++) {
    var order = orders[i];
    // getting cache via key
    var orderline = this.orders_history_cache.get_node(order.id);
    var lines_table = this.orders_history_cache.get_node(order.id + '_table');
    /* here we check for the presence of cache among existing data
     if there is no cache, then we render elements and save into cache
     if the cache exists, we just use it
    */
    if ((!orderline) || (!lines_table)) {
      // rendering of elements may take time
      var orderline_html = QWeb.render('OrderHistory', {widget: this, order:order});
      orderline = document.createElement('tbody');
      lines_table = document.createElement('tr');
      var $td = document.createElement('td');
      if (order.lines) {
        $td.setAttribute("colspan", 8);
      }
      lines_table.classList.add('line-element-hidden');
      lines_table.classList.add('line-element-container');

      var line_data = this.get_order_line_data(order);
      var $table = this.render_lines_table(line_data);

      $td.appendChild($table);
      lines_table.appendChild($td);

      orderline.innerHTML = orderline_html;
      orderline = orderline.childNodes[1];
      //save the result into cache
      this.orders_history_cache.cache_node(order.id, orderline);
      this.orders_history_cache.cache_node(order.id + '_table', lines_table);
    }
    contents.appendChild(orderline);
    contents.appendChild(lines_table);
  }
},
```

## JS access and inheritance

### **action\_button**

Here you will find explanation of how to get/inherit action\_button POS objects.

For example we have definition in this file:

```
odoo.define('pos_reprint.pos_reprint', function (require) {
...
screens.define_action_button({
    'name': 'guests',
    'widget': TableGuestsButton,
    'condition': function()
```

This defenition doesn't return class ReprintButton. So, we cannot inherit it in a usual way.

In order to reach that object we need get instance of it using `gui`. Then we can inherit it

To make clear what this is like look up example where guests number button renderings:

```
this.gui.screen_instances['products'].action_buttons['guests'].renderElement();
```

While you can make call and even replace function with new one, you are not able to make inheritance via `extend` or `include` functions. It's because we cannot reach Class and only get access to instance of that class.

This kind of approach make sense only for those widgets:

```
DiscountButton
ReprintButton
TableGuestsButton
SubmitOrderButton
OrderlineNoteButton
PrintBillButton
SplitbillButton
set_fiscal_position_button
```

### **screen\_classes**

To create new screen widget (via the `extend()` method) or to modify existing screen widget (via the `include()` method) you need the target class. Usually you can get this class using following code:

```
odoo.define('module_name.file_name', function (require) {
"use strict";

var screens = require('point_of_sale.screens');

screens.OrderWidget.include({
```

But it is available only for widgets that are returned by main function in the file “`point_of_sale/static/src/js/screens.js`”.

#### **List of the screens:**

- ReceiptScreenWidget
- ActionButtonWidget
- define\_action\_button
- ScreenWidget

- PaymentScreenWidget
  - OrderWidget
  - NumpadWidget
  - ProductScreenWidget
  - ProductListWidget

In other cases you can get targeted screen widget class using following code:

```
odoo.define('module_name.file_name', function (require) {
"use strict";

var gui = require('point_of_sale.gui');

gui.Gui.prototype.screen_classes.filter(function(el) { return el.name == 'clientlist' })
[0].widget.include({
    ...
});
```

List of screens available via screen\_classes:

```
gui.define_screen({name: 'scale', widget: ScaleScreenWidget});
gui.define_screen({name: 'products', widget: ProductScreenWidget});
gui.define_screen({name: 'clientlist', widget: ClientListScreenWidget});
gui.define_screen({name: 'receipt', widget: ReceiptScreenWidget});
gui.define_screen({name: 'payment', widget: PaymentScreenWidget});
gui.define_screen({name: 'bill', widget: BillScreenWidget});
gui.define_screen({'name': 'splitbill', 'widget': SplitbillScreenWidget,
gui.define_screen({'name': 'floors', 'widget': FloorScreenWidget,
```

## 2.11 Access

## 2.11.1 Security tutorial

## **Resources:**

- [http://odoo-docs.readthedocs.org/en/latest/04\\_security.html](http://odoo-docs.readthedocs.org/en/latest/04_security.html)
  - <https://www.odoo.com/documentation/9.0/howtos/backend.html#security>
  - <https://www.odoo.com/documentation/9.0/reference/security.html>

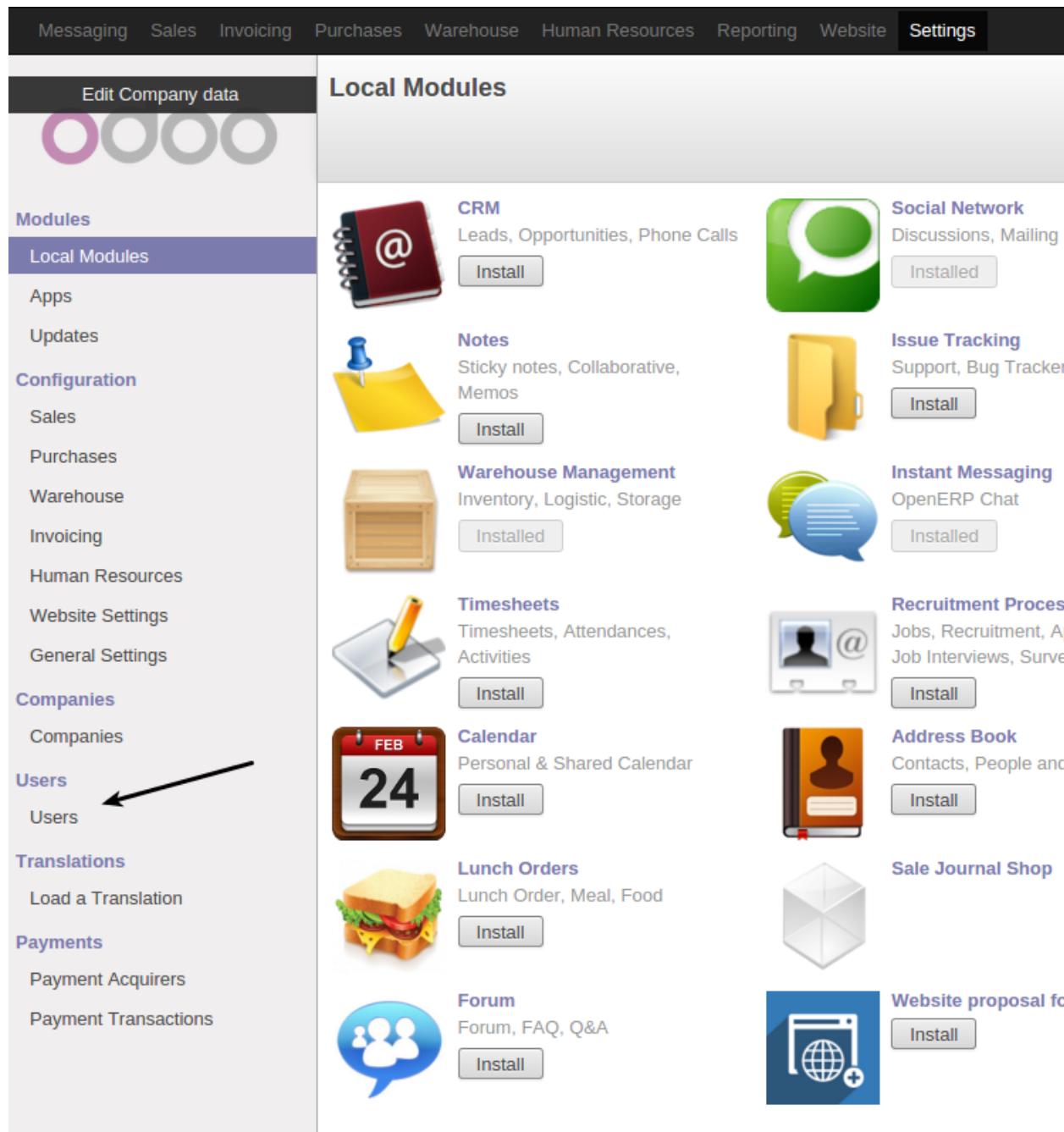
Odoo is very flexible on the subject of security. We can control what users can do and what they cannot on different levels. Also we can control independently each of the four basic operations: read, write, create, unlink. I.e. allow only read, allow only create, grant permission to create or delete only.

**On fields/menu level we can:**

- hide fields or menus for some users and show them for others
  - make fields readonly for some users and make them editable for others
  - show different variants to pick on the Selection fields for different users

On the fields level of security `res.users` and `res.groups` models are used. These models relate to each other as many2many. This means that a user can be a member of many groups and one group can be assigned to many users.

One example of how we can hide menu in regard to current user's groups is the following.



On the picture above in Settings / Users we can see only Users menu. We know that there should be Groups menu also. Let Us see in `./openerp/addons/base/res/res_users_view.xml` on the point of how menuitem can be hidden.

```
<record id="action_res_groups" model="ir.actions.act_window">
    <field name="name">Groups</field>
    <field name="type">ir.actions.act_window</field>
    <field name="res_model">res.groups</field>
    <field name="view_type">form</field>
    <field name="help">A group is a set of functional areas that will be assigned to  
the
    </field>

```

(continues on next page)

(continued from previous page)

```

user in order to give them access and rights to specific applications and tasks in
the system. You can create custom groups or edit the ones existing by default
in order to customize the view of the menu that users will be able to see. Whether
they can have a read, write, create and delete access right can be managed from
here.
</field>
</record>
<menuitem action="action_res_groups" id="menu_action_res_groups" parent="base.menu_
↳users"
groups="base.group_no_one"/>

```

The groups attribute in the `menuitem` element shows us that only the members of `base.group_no_one` group can see the Groups menu item. The `base.group_no_one` xmlid is defined in the `./openerp/addons/base/security/base_security.xml` as follows.

```

<record model="res.groups" id="group_erp_manager">
    <field name="name">Access Rights</field>
</record>
<record model="res.groups" id="group_system">
    <field name="name">Settings</field>
    <field name="implied_ids" eval="[(4, ref('group_erp_manager'))]"/>
    <field name="users" eval="[(4, ref('base.user_root'))]"/>
</record>

<record model="res.groups" id="group_user">
    <field name="name">Employee</field>
    <field name="users" eval="[(4, ref('base.user_root'))]"/>
</record>

<record model="res.groups" id="group_multi_company">
    <field name="name">Multi Companies</field>
</record>

<record model="res.groups" id="group_multi_currency">
    <field name="name">Multi Currencies</field>
</record>

<record model="res.groups" id="group_no_one">
    <field name="name">Technical Features</field>
</record>

<record id="group_sale_salesman" model="res.groups">
    <field name="name">User</field>
</record>
<record id="group_sale_manager" model="res.groups">
    <field name="name">Manager</field>
    <field name="implied_ids" eval="[(4, ref('group_sale_salesman'))]"/>
</record>

```

Here we can see the `group_no_one` along with the other base groups. Note that `group_no_one` has Technical Features name. Let us include our user in the Technical Features group. Since we have no access to the Groups menu item, the only way we can do it is from the Users menu item. See the picture below.

The screenshot shows the Odoo web interface for managing users. On the left, a sidebar lists various modules: Local Modules, Apps, Updates, Configuration (Sales, Purchases, Warehouse, Invoicing, Human Resources, Website Settings, General Settings), Companies (Companies), Users (Users, Translations, Payments), and a few others like Multi Companies and Technical Features.

The main content area displays the 'Administrator' user profile. At the top, there are 'Edit' and 'Create' buttons, and a 'More' dropdown. Below that is a placeholder image for a profile picture. The user information shows 'Administrator' as the name and 'admin' as the login.

The 'Active' status is checked. Below the user info, there are tabs for 'Access Rights' and 'Preferences'. The 'Access Rights' tab is selected, showing the following application access rights:

Application	Access Right
Sales	Manager
Warehouse	Manager
Accounting & Finance	Invoicing & Payments
Purchases	Manager
Human Resources	Manager
Website	Manage Website and qWeb view
Sharing	User
Administration	Settings

Below the access rights, there are sections for 'Usability' and 'Other'. Under 'Usability', there is a 'Multi Companies' checkbox which is unchecked. Under 'Other', there are checkboxes for 'Booking Staff', 'Front Desk', 'Public', 'Website Comments', 'Contact Creation', 'Portal', and 'Sales / See taxes'. The 'Contact Creation' checkbox is checked and highlighted with a black arrow.

Check the Technical Features box and reload odoo. Now we can see the Groups menu item!

**Local Modules**

<b>CRM</b> Leads, Opportunities, Phone Calls <i>crm</i>	<b>Social Network</b> Discussions, Mailing <i>mail</i>
<b>Online Billing</b> Send Invoices and Track Payments <i>account_voucher</i>	<b>Point of Sale</b> Touchscreen Interfac <i>point_of_sale</i>
<b>Project Management</b> Projects, Tasks <i>project</i>	<b>Notes</b> Sticky notes, Collabo Memos <i>note</i>
<b>Issue Tracking</b> Support, Bug Tracker, Helpdesk <i>project_issue</i>	<b>Accounting and Fin</b> Financial and Analyti <i>account_accountant</i>
<b>Survey</b> Create surveys, collect answers and print statistics <i>survey</i>	<b>Sales Management</b> Quotations, Sales Or Invoicing <i>sale</i>
<b>Warehouse Management</b> Inventory, Logistic, Storage <i>stock</i>	<b>Instant Messaging</b> OpenERP Chat <i>im_chat</i>

From Settings / Users / Groups we can see a list of existing groups. Here we also can assign users for groups.

### Hide fields

In the `./openerp/addons/base/res/res_users_view.xml` we can see the `view_users_simple_form` view. Note here that the `company_id` field is visible only for members of the `base.group_multi_company` group.

```
<!-- res.users -->
<record id="view_users_simple_form" model="ir.ui.view">
    <field name="name">res.users.simplified.form</field>
    <field name="model">res.users</field>
```

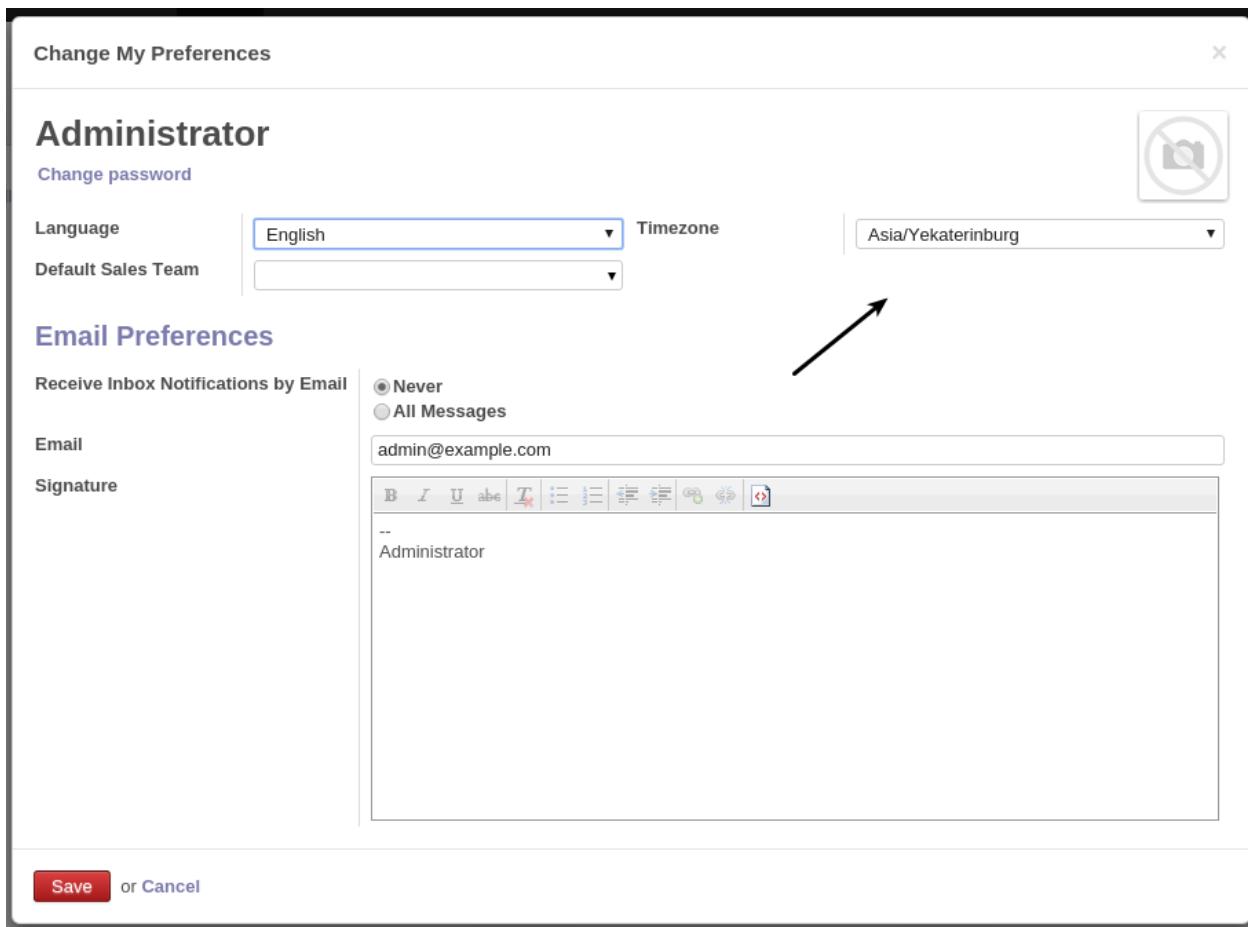
(continues on next page)

(continued from previous page)

```

<field name="priority">1</field>
<field name="arch" type="xml">
    <form string="Users">
        <sheet>
            <field name="id" invisible="1"/>
            <div class="oe_form_box_info oe_text_center" style="margin-bottom: 10px" attrs="{'invisible': [('id', '>', 0)]}">
                You are creating a new user. After saving, the user will receive an invite email containing a link to set its password.
            </div>
            <field name="image" widget='image' class="oe_avatar oe_left" options='{"preview_image": "image_medium"}'>
                <div class="oe_title">
                    <label for="name" class="oe_edit_only"/>
                    <h1><field name="name"/></h1>
                    <field name="email" invisible="1"/>
                    <label for="login" class="oe_edit_only" string="Email Address"/>
                    <h2>
                        <field name="login" on_change="on_change_login(login)" placeholder="email@yourcompany.com"/>
                    </h2>
                    <label for="company_id" class="oe_edit_only" groups="base.group_multi_company"/>
                        <field name="company_id" context="{'user_preference': 0}" groups="base.group_multi_company"/>
                    </div>
                    <group>
                        <label for="groups_id" string="Access Rights" attrs="{'invisible': [('id', '>', 0)]}">
                            <div attrs="{'invisible': [('id', '>', 0)]}">
                                <field name="groups_id" readonly="1" widget="many2many_tags" style="display: inline;"> You will be able to define additional access rights by editing the newly created user under the Settings / Users menu.
                            </div>
                            <field name="phone"/>
                            <field name="mobile"/>
                            <field name="fax"/>
                        </group>
                    </sheet>
                </form>
            </field>
        </record>
    
```

Our current user is Administrator. By default he is not a member of the `base.group_multicompany` group. That is why the `company_id` isn't visible for him on the form.



#### Model records:

- restrict access to specified subset of records in model

#### Model:

- restrict access to all records of model

### 2.11.2 Superuser rights

Administrator, i.e. user with id 1 (SUPERUSER\_ID), has exceptions about access rights.

#### `ir.model.access`

If some model doesn't have records in `ir.model.access` (*Access Rules*), then only Administrator has access to that model.

See also:

- `ir.model.access`
- `ir.rule`

### 2.11.3 Video Lessons

- (Russian)

## 2.12 Hooks

### 2.12.1 post\_load

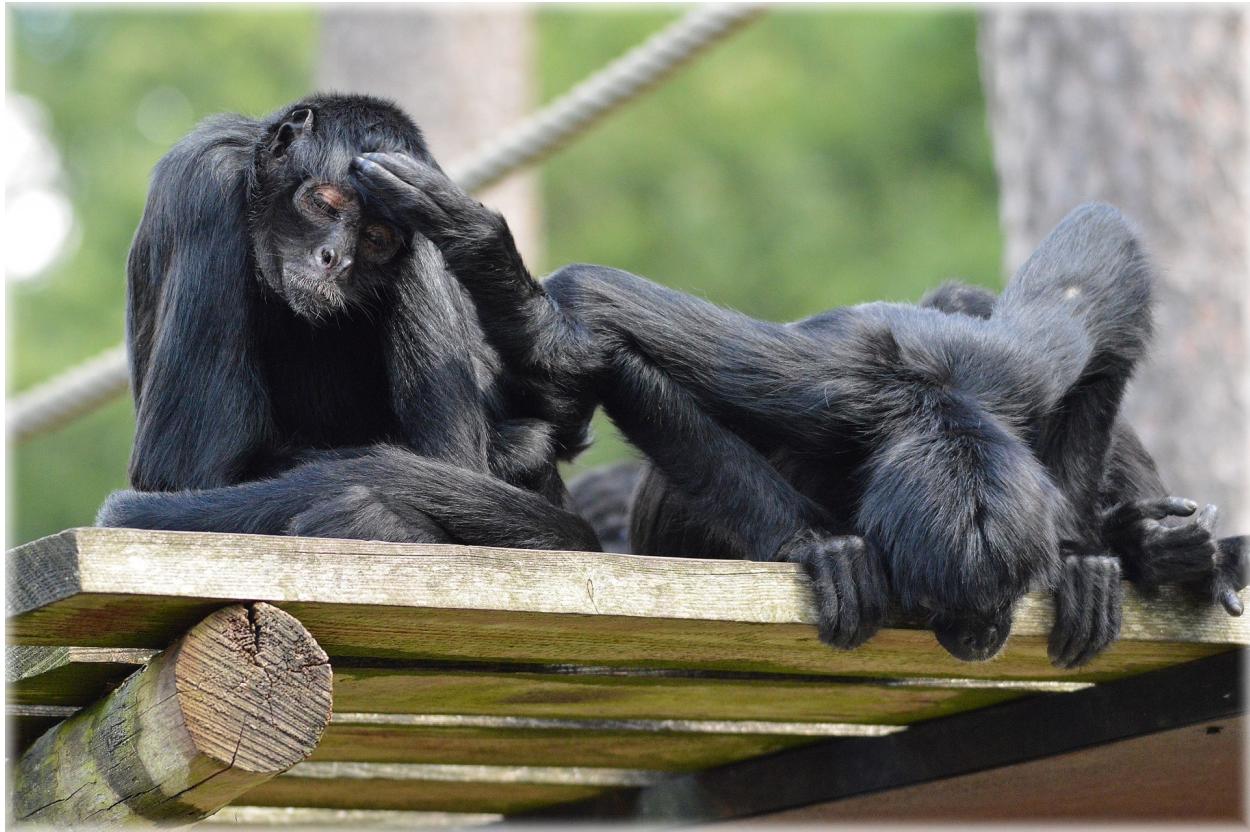
- *What do we know from comments in odoo source?*
- *What is it actually for?*
- *Example of monkey patch in odoo*
- *Why shall we use post\_load to apply monkey patch?*
- *How to use post\_load?*
- *Example?*
- *Something else we need to know?*
- *Other usage of post\_load?*

#### What do we know from comments in odoo source?

```
# Call the module's post-load hook. This can done before any model or
# data has been initialized. This is ok as the post-load hook is for
# server-wide (instead of registry-specific) functionalities.
```

#### What is it actually for?

For Monkey patches



### Example of monkey patch in odoo

```
from odoo import tools

def new_image_resize_images(*args, **kwargs):
    ...

tools.image_resize_images = new_image_resize_images
```

### Why shall we use `post_load` to apply monkey patch?

---

**Note:** Since odoo 12 monkey patch could be applied without `post_load`, but it's still recommended to use it to be sure.

---

Because otherwise monkey patch will be applied every time it is available in addons path. It happens because odoo loads python files of a module if there is a static folder in the module (no matter if the module is installed or not – see `load_addons` method in `http.py` file of odoo source).

### How to use `post_load`?

You need to define a function available in `__init__.py` file of the module. Then set that function name as value of "post\_load" attribute in module manifest.

## Example?

Sure. E.g. from `telegram` module.

In `__openerp__.py`

```
...
"post_load": "telegram_worker",
"pre_init_hook": None,
"post_init_hook": None,
"installable": True,
"auto_install": False,
"application": True,
}
```

In `__init__.py`

```
from odoo.service.server import PreforkServer

...

def telegram_worker():
    # monkey patch
    old_process_spawn = PreforkServer.process_spawn

    def process_spawn(self):
        old_process_spawn(self)
        while len(self.workers_telegram) < self.telegram_population:
            # only 1 telegram process we create.
            self.worker_spawn(WorkerTelegram, self.workers_telegram)

    PreforkServer.process_spawn = process_spawn
    old_init = PreforkServer.__init__

    def __init__(self, app):
        old_init(self, app)
        self.workers_telegram = {}
        self.telegram_population = 1
    PreforkServer.__init__ = __init__
```

## Something else we need to know?

Yes.

Additionally, if you need to apply monkey patch before any other initialisation, the module has to be added to `server_wide_modules` parameter.

## Other usage of `post_load`?

In case of extending pos-box modules (e.g. `hw_escpos`), you probably need to use `post_load`, because importing `hw_escpos` from your module runs posbox specific initialisation.

Example from `hw_printer_network` module:

In `__manifest__.py`

```

    ...
    "post_load": "post_load",
    "pre_init_hook": None,
    "post_init_hook": None,
    "installable": True,
    "auto_install": False,
    "application": True,
}
```

In `__init__.py`

```
def post_load():
    from . import controllers
```

In `controllers/hw_printer_network_controller.py`

```

# first reason of using post_load
from odoo.addons.hw_escpos.escpos import escpos
import odoo.addons.hw_escpos.controllers.main as hw_escpos_main

...
# second reason - monkey patch:
driver = UpdatedEscposDriver()
hw_escpos_main.driver = driver
```

## 2.13 Source Diving

Source Diving is a way to find answers to your questions.

### 2.13.1 Source Diving Cases

This section contains live examples of source diving.

Each case contains problem description and possible solutions. Use problems as exercises and solutions as manual.

#### Case: “Transformed the method”

##### Context

When porting module `mail_move_message` in the file `static/src/js/mail_move_message.js` there is a method `session.web.form.FormOpenPopup(this)`.

##### Problem

In 9.0 not found such object. What object would be the analogue of the object? What you need to do to find this object?

## Solution

Possible solution

## Guidelines

Use template below for new cases

```
=====
CASE NAME
=====

Context
=====

What we have. E.g. some module, or out-of-box odoo version 8.0
* LINK1
* LINK2

Problem
=====

What we need to do. E.g. port module to 9.0
* LINK1
* LINK2

Solution
=====

:doc:`Possible solution <./answers/CASE_NAME>`
```

### 2.13.2 Overview: “Transformed the method”

Quite often when porting a module from 8.0 to 9.0 there is a situation, when 8.0 is a object, but there is no 9.0. And it is not clear - it is outdated and it was removed or it was renamed. In very advanced cases, an object can be renamed and changed almost beyond recognition.

To search you need to take several steps:

1. The default view that such an object exist, but it was renamed.
2. Look, what makes this object.
3. Search by name of methods that contains the given object, excluding common words (for example, init, start, destroy...).
4. If the result is not found that search by unique keywords which can be found by bringing the object.
5. If anything gave no results, then maybe the object is deleted as obsolete.

*Case*

Possible solution

## 2.14 Lint

### 2.14.1 Installation

```
# install autopep8
sudo pip install --upgrade autopep8

# install oca-autopep8
git clone https://github.com/OCA/maintainer-tools.git
cd maintainer-tools
sudo python setup.py install

# install autoflake
sudo pip install --upgrade autoflake

# install fixmyjs
sudo npm install fixmyjs -g
# increase max errors to be fixed (otherwise script stops)
echo '{"maxerr": 1000}' > ~/.jshintrc
```

### 2.14.2 Common lints

```
EXCLUDE_FILES=".\\(svg\\|gif\\|png\\|jpg\\)$"
# fix line break symbols
cd /path/to/MODULE_NAME
find * -type f | grep -v $EXCLUDE_FILES | xargs sed -i 's/\\r//g'

# add line break to the end of file
find * -type f | grep -v $EXCLUDE_FILES | xargs sed -i '$a\'

# trim trailing whitespaces
find * -type f | grep -v $EXCLUDE_FILES | xargs sed -i 's/[ \\t]*$//g'

# Replacement button 'Tab' on 4 button 'Space':
find . -type f -name '*.xml' -or -name '*.py' -or -name '*.js'| xargs sed -i 's/\\t/ ↵ /g'
```

#### Fixing python lints in odoo

##### All versions

```
# PEP8 for py-files:
autopep8 --in-place -r --aggressive --aggressive --ignore E501 ./

# fix CamelCase
oca-autopep8 -ri --select=CW0001 .

# Replacement (relative-import)
find . -type f -name '__init__.py' | xargs sed -i 's/^import/from . import/g'
#find . -type f -name '__init__.py' | xargs sed -i 's/^import controllers/from . ↵ import controllers/g'
#find . -type f -name '__init__.py' | xargs sed -i 's/^import models/from . import ↵ models/g'
```

(continues on next page)

(continued from previous page)

```
# remove unused imports
autoflake --in-place -r --imports=openerp,openerp.http.request,openerp.SUPERUSER_ID,
→openerp.addons.base.ir.ir_qweb,openerp.exceptions.ValidationError,openerp.fields,
→openerp.api.openerp.models,openerp.osv.fields,openerp.osv.api,telebot,lxml,werkzeug,
→MySQLdb.cursors,cStringIO.StringIO,werkzeug.utils,pandas.merge,pandas.DataFrame,
→werkzeug.wsgi.wrap_file,werkzeug.wsgi,werkzeug.wsgi.wrap_file,openerp.exceptions,
→openerp.tools.DEFAULT_SERVER_DATETIME_FORMAT ./

# remove prints
find . -type f -name '*.py' | xargs sed -i 's/^(\ *)\((print .*)\)/\1# \2/g'

#Fix comments:
find . -type f -name '*.py' | xargs sed -i -e 's/ #\([^\ ]\)/ # \1/g'

# Correction is rights for run:
find -iname '*.py' | xargs chmod -x
```

## Odoo 10-

```
# Addition of the first row (coding) in py-files
find -iname '*.py' | grep -v "__init__.py" | xargs grep -rlP 'coding: *utf-8' | xargs
→sed -i 'ls/^/# -*- coding: utf-8 -*-\n/'
```

## @api.one -> @api.multi

```
# Note. This solution doesn't work on methods that call super (e.g. write, create,
→methods) or has to return value
# Note. This solution doesn't handle properly methods with kwargs
find . -type f -name '*.py' | xargs perl -i -p0e 's/'\n
'@api.one\n'\n
'    def ([^()]*)\($self, ([^()]*)\):/\n
'@api.multi\n'\n
'    def $1($self, $2):\n'
'        for r in self:\n'
'            r.$1_one($2)\n'
'        return True'\n
'\n\n'
'    \@api.multi\n'\n
'    def $1_one($self, $2):\n'
'        self.ensure_one()/g'

find . -type f -name '*.py' | xargs perl -i -p0e 's/'\n
'@api.one\n'\n
'    def ([^()]*)\($self\):/\n
'@api.multi\n'\n
'    def $1($self):\n'
'        for r in self:\n'
'            r.$1_one()\n'
'        return True'\n
```

(continues on next page)

(continued from previous page)

```
'\n'\
'\n'\
'    @api.multi\n'\
'    def $1_one(self):\n'\
'        self.ensure_one()/g'
```

## Fixing Javascript lints in odoo

```
#lint for js:
fixmyjs --legacy --config ~/.jshintrc ./
```

## Fixing rst lints in odoo

```
# Correction is links in rst-files
#`_  ->  __
find . -type f -name '*.rst' | xargs sed -i '`_(?!_)`__/g'
```

## Fixing xml lints in odoo

```
# xml-deprecated-tree-attribute
find . -type f -name '*.xml' | xargs sed -i 's/\(\<tree.*\)\ string="[^"]*/\1/g'
```

## 2.15 Other

### 2.15.1 Dynamic records

While *XML* allows you create only *static* records, there is a way to create record dynamically via python code. You need dynamic records, for example, to add support both for enterprise and community releases or to add some records to each company in database etc.

There several ways to execute code on installation:

- TODO
- TODO
- TODO

The problem with dynamic records is that odoo considers such records as ones, which were in xml files, but now deleted. It means that odoo will delete such dynamic records right after updating. There are two ways to resolve it.

#### **noupdate=False**

Simply add update=True to your `ir.model.data` record:

```

debt_account = registry['account.account'].create(cr, SUPERUSER_ID, {
    'name': 'Debt',
    'code': 'XDEBT',
    'user_type_id': registry.get('ir.model.data').get_object_reference(cr, SUPERUSER_ID, 'account', 'data_account_type_current_assets')[1],
    'company_id': company.id,
    'note': 'code "XDEBT" should not be modified as it is used to compute debt',
})
registry['ir.model.data'].create(cr, SUPERUSER_ID, {
    'name': 'debt_account_' + str(company.id),
    'model': 'account.account',
    'module': 'pos_debt_notebook',
    'res_id': debt_account,
    'noupdate': True, # If it's False, target record (res_id) will be removed while module update
})

```

## noupdate=True

If for some reason you cannot use noupdate=False, you can use following trick.

Here is the example from `web_debranding` module. To create records in `ir.model.data` we use name `_web_debranding`. Then odoo will consider such records as belonging to another module (`_web_debranding`) and will not delete them. But it also means, that odoo will not delete them after uninstalling. For later case, we need to use `uninstall_hook`.

## Contents

- *Dynamic records*
  - `noupdate=False`
  - `noupdate=True`
    - \* `python file`
    - \* `yaml file`
    - \* `__openerp__.py`
    - \* `__init__.py`

## python file

```

from openerp import SUPERUSER_ID, models, tools, api

MODULE = '_web_debranding'

class view(models.Model):
    _inherit = 'ir.ui.view'

    def _create_debranding_views(self, cr, uid):
        self._create_view(cr, uid, 'menu_secondary', 'web.menu_secondary', '')

```

(continues on next page)

(continued from previous page)

```

<xpath expr="//div[@class='oe_footer']" position="replace">
    <div class="oe_footer"></div>
</xpath>'')

```

```

def _create_view(self, cr, uid, name, inherit_id, arch, nouupdate=False, type='qweb'
↳) :
    registry = self.pool
    view_id = registry['ir.model.data'].xmlid_to_res_id(cr, SUPERUSER_ID, "%s.%s"
↳ (MODULE, name))
    if view_id:
        registry['ir.ui.view'].write(cr, SUPERUSER_ID, [view_id], {
            'arch': arch,
        })
    return view_id

try:
    view_id = registry['ir.ui.view'].create(cr, SUPERUSER_ID, {
        'name': name,
        'type': type,
        'arch': arch,
        'inherit_id': registry['ir.model.data'].xmlid_to_res_id(cr, SUPERUSER_
↳ ID, inherit_id, raise_if_not_found=True)
    })
except:
    import traceback
    traceback.print_exc()
return
registry['ir.model.data'].create(cr, SUPERUSER_ID, {
    'name': name,
    'model': 'ir.ui.view',
    'module': MODULE,
    'res_id': view_id,
    'noupdate': nouupdate,
})
return view_id

```

**yaml file**

```

-
!python {model: ir.ui.view}: |
    self._create_debranding_views(cr, uid)

```

**\_\_openerp\_\_.py**

```

'uninstall_hook': 'uninstall_hook',
'data': [
    'path/to/file.yml'
]

```

## \_\_init\_\_.py

```
from openerp import SUPERUSER_ID

MODULE = '_web_debranding'
def uninstall_hook(cr, registry):
    registry['ir.model.data']._module_data_uninstall(cr, SUPERUSER_ID, [MODULE])
```

## 2.15.2 Odoo database

### Many to many

For every *many to many* field odoo creating new relations table for example *pos\_multi\_rel* with *\_rel* postfix.

## 2.15.3 Odoo way of shaman

### What to do if something not work but should to

1. Refresh page
2. Update module
3. Check openerp file **depends**, **demo** and other important fields
4. Check odoo config you use to run odoo. Especially addons paths
5. Uninstall and install again modules in depends
6. Clean browser cache
7. Carefully check logs. Look up if needed files loaded or not. May be some errors.
8. Create new base and install all modules.

# CHAPTER 3

---

## Debugging

---

This section describes how to find the reason of existing problem.

### 3.1 Terminal logs

Logs from **terminal** (*in development environment*) or **log file** (*in production environment*) are primary source to find the reason of a problem.

To control output level use `--log-handler`

#### 3.1.1 Output format

Default format is as following:

```
%(asctime)s %(pid)s %(levelname)s %(dbname)s %(name)s: %(message)s
```

```
2017-12-23 10:32:59,388 13 INFO point_of_sale-10 werkzeug: 172.17.0.1 -- [23/Dec/  
->2017 10:32:59] "POST /web/webclient/translations HTTP/1.0" 200 -  
asctime_____ PID LEVEL DB_NAME_____ NAME_____ MESSAGE_____  
->_____
```

#### Name

*Name* is argument of creation `_logger` object. Usually it's equal to

```
_logger = logging.getLogger(__name__)
```

i.e. equal to package name

## PID

*PID* is a process ID. E.g. ID of one of *worker* or *cron process*

## Message

*Message* is anything passing to one of logging method, e.g. `_logger.info(Message)`

## 3.2 Browser's Console

Browser's console (short name: *console*) may contain userfull logs about client part.

To open console Click F12 in browser.

## 3.3 Sources tab at Browser's dev tools

Allows you to check which client side files are loaded and which are not. To do this:

1. Turn on *debug mode (with assets)*
2. Open Developer tools (F12), go to the Sources tab and reload page.
3. Open left panel (if it is not open yet) and search interested app.

Example: *Missing dependencies error in console*

## 3.4 Network tab at Browser's dev tools

Sometimes error are not printed neither in *Terminal* nor in *Console*. Then you can try to find some usefull information at Network tab of browser's developer tools.

### 3.4.1 Response value

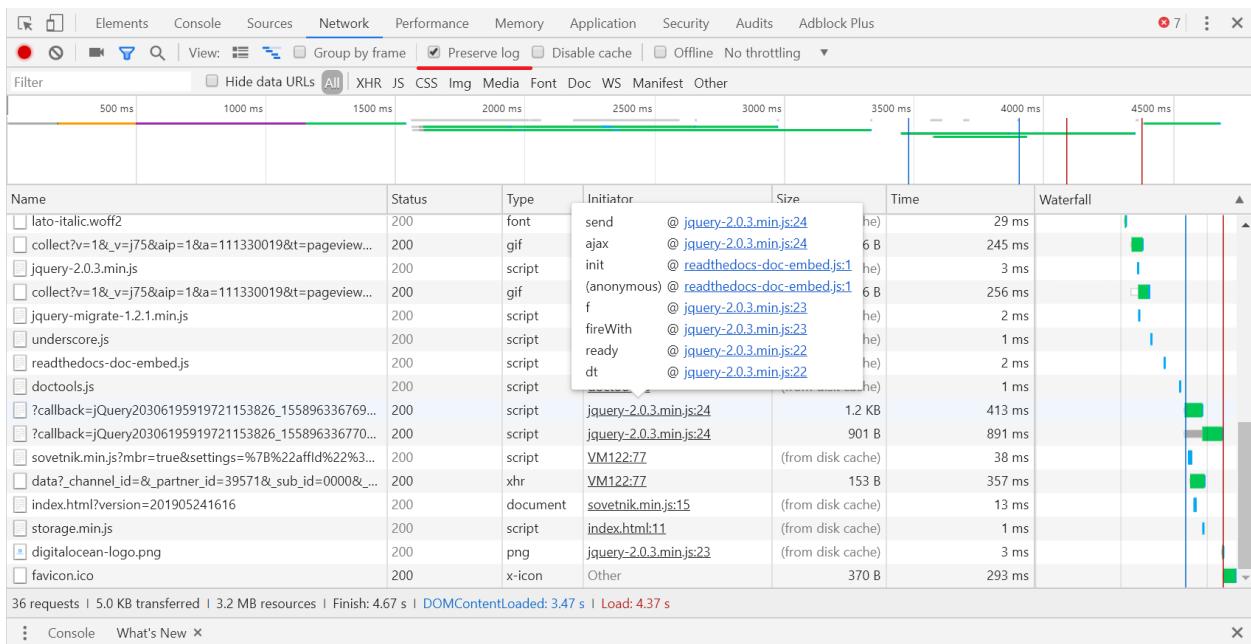
To see Response click on the request line and then navigate to Response tab.

### 3.4.2 Who made http request

Suppose we want to know which part of our script initiate the request. To do that put mouse pointer above initiator column's element.

### 3.4.3 Preserve log

Ticking the **Preserve log** checkbox will save your console output across page refreshes and closing / reopening Browser's dev tools. Console history will only clear when the tab is closed or you manually clear the Console.



**Note:** To see original odoo js files i.e. not minimized versions, open odoo in *debug mode (with assets)* first

## 3.5 QWeb

The javascript QWeb implementation provides a few debugging hooks:

**t-log** takes an expression parameter, evaluates the expression during rendering and logs its result with `console.log`:

```
<t t-set="foo" t-value="42"/>
<t t-log="foo"/>
```

will print 42 to the console

**t-debug** triggers a debugger breakpoint during template rendering:

```
<t t-if="a_test">
  <t t-debug="">
</t>
```

will stop execution if debugging is active (exact condition depend on the browser and its development tools)

**t-js** the node's body is javascript code executed during template rendering. Takes a `context` parameter, which is the name under which the rendering context will be available in the `t-js`'s body:

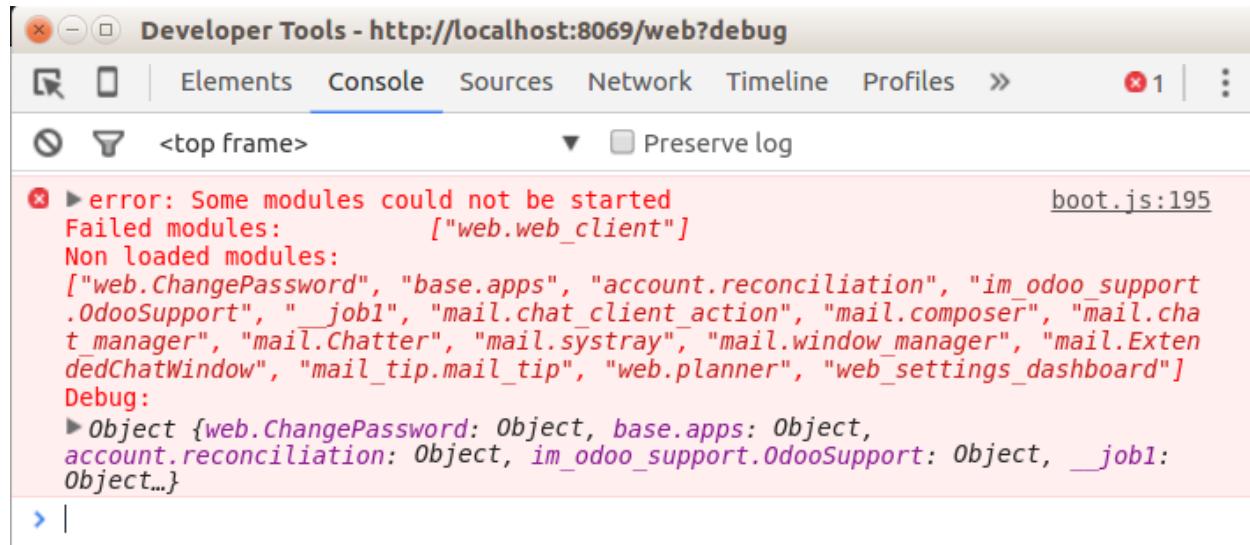
```
<t t-set="foo" t-value="42"/>
<t t-js="ctx">
  console.log("Foo is", ctx.foo);
</t>
```

Source

## 3.6 Typical errors

### 3.6.1 Error: Failed modules

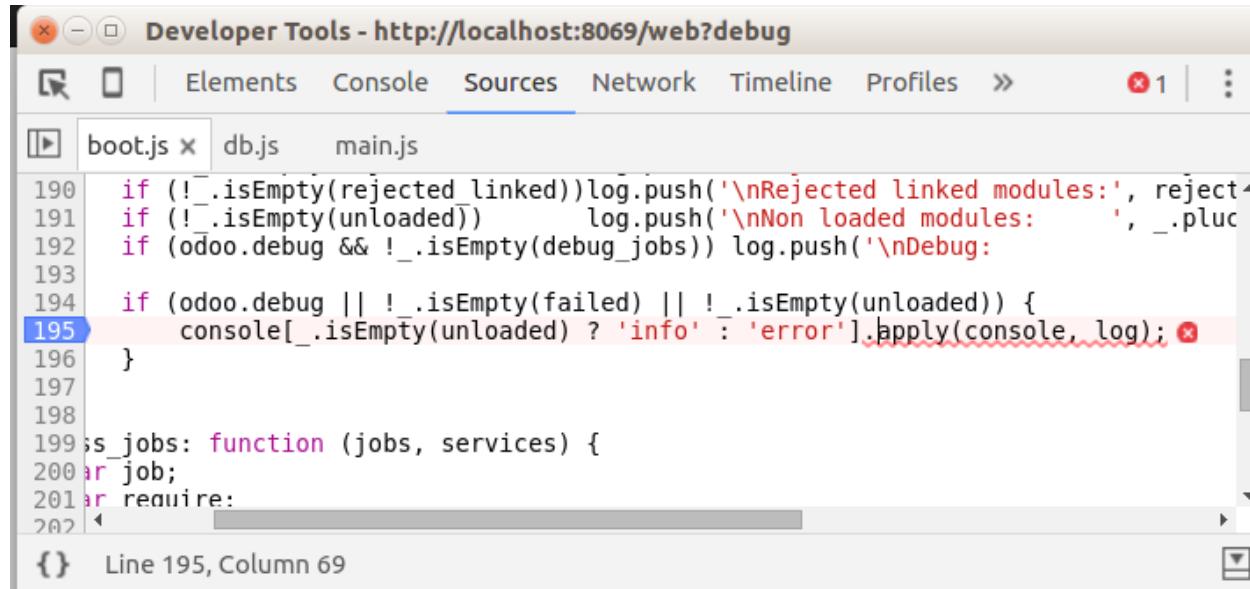
If into server console no errors but boot.js raise exception that find out reason error next steps:



```
Developer Tools - http://localhost:8069/web?debug
Elements Console Sources Network Timeline Profiles > ① : 
<top frame> ▾ Preserve log

error: Some modules could not be started
Failed modules: ["web.web_client"]
Non loaded modules:
["web.ChangePassword", "base.apps", "account.reconciliation", "im_odoo_support.OdooSupport", "__job1", "mail.chat_client_action", "mail.composer", "mail.chat_manager", "mail.Chatter", "mail.systray", "mail.window_manager", "mail.Exten dedChatWindow", "mail_tip.mail_tip", "web.planner", "web_settings_dashboard"]
Debug:
▶ Object {web.ChangePassword: Object, base.apps: Object,
account.reconciliation: Object, im_odoo_support.OdooSupport: Object, __job1: Object...}
> |
```

1. Go to error line into boot.js.
2. Turn on breakpoint.



```
Developer Tools - http://localhost:8069/web?debug
Elements Console Sources Network Timeline Profiles > ① : 
boot.js x db.js main.js

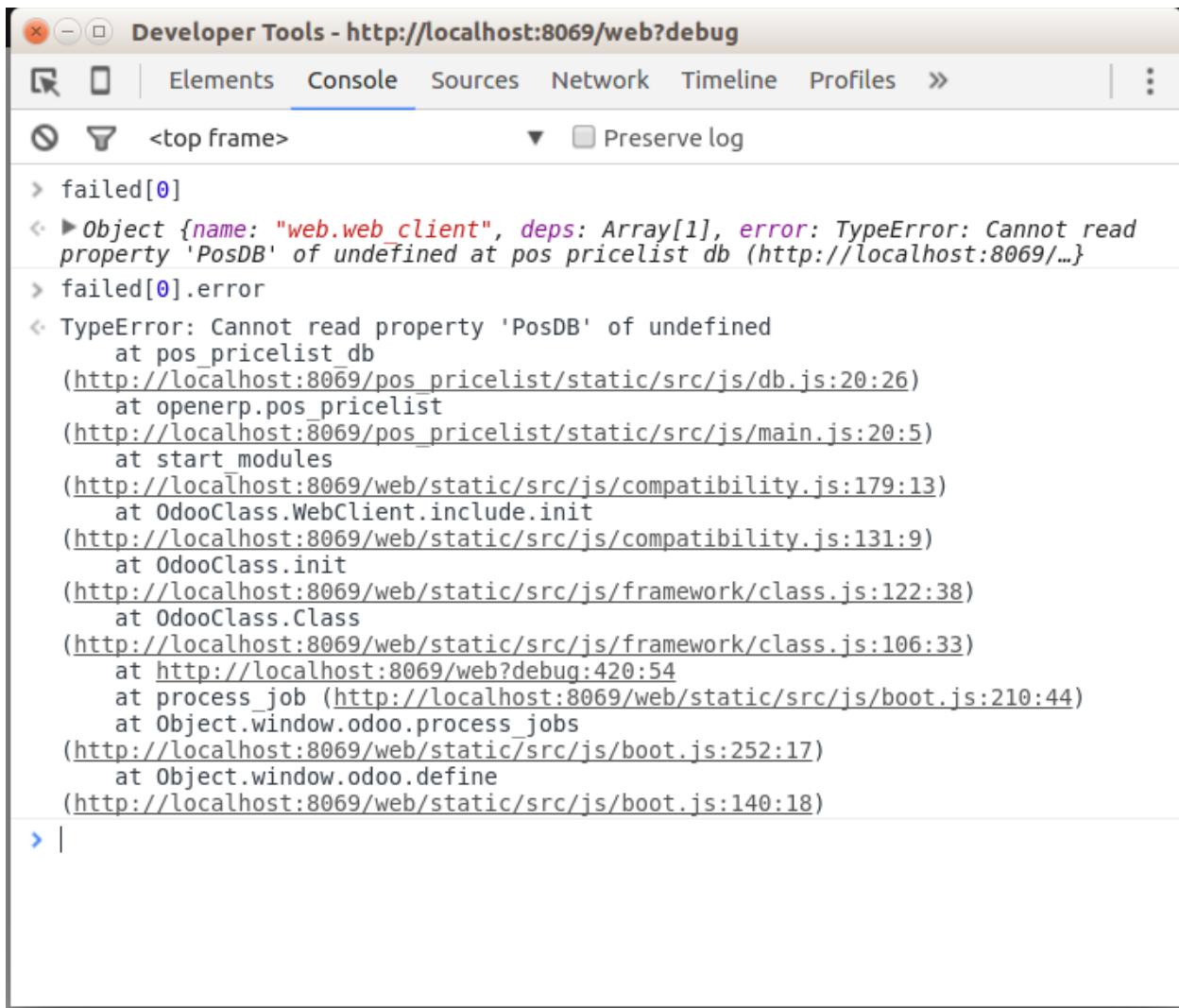
190 if (!_.isEmpty(rejected_linked)) log.push('\nRejected linked modules:', reject^
191 if (!_.isEmpty(unloaded)) log.push('\nNon loaded modules: ', __pluc
192 if (odoo.debug && !_.isEmpty(debug_jobs)) log.push('\nDebug:
193
194 if (odoo.debug || !_.isEmpty(failed) || !_.isEmpty(unloaded)) {
195   console[_.isEmpty(unloaded) ? 'info' : 'error'].apply(console, log); x
196 }
197
198
199 ss_jobs: function (jobs, services) {
200   var job;
201   var require:
202 }

{} Line 195, Column 69
```

3. Rerun script (click F5)
4. When script stop on error line move to console.
5. Type command:

```
failed[0].error
```

6. To receive the output



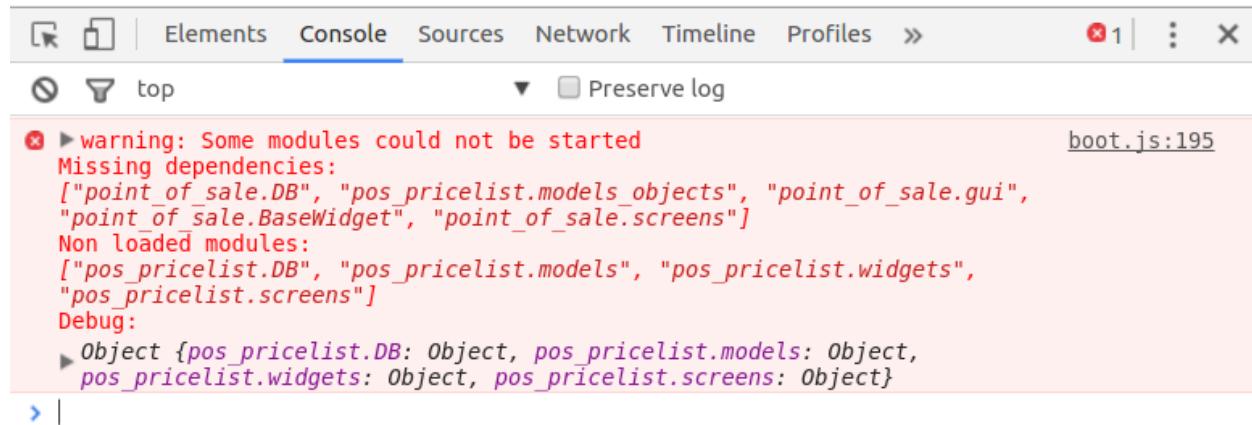
The screenshot shows the Chrome Developer Tools with the 'Console' tab selected. The title bar says 'Developer Tools - http://localhost:8069/web?debug'. The console output shows an error stack trace:

```
> failed[0]
< ► Object {name: "web.web_client", deps: Array[1], error: TypeError: Cannot read property 'PosDB' of undefined at pos pricelist db (http://localhost:8069/...)}
> failed[0].error
< TypeError: Cannot read property 'PosDB' of undefined
    at pos_pricelist_db
    (http://localhost:8069/pos_pricelist/static/src/js/db.js:20:26)
    at openerp.pos_pricelist
    (http://localhost:8069/pos_pricelist/static/src/js/main.js:20:5)
    at start_modules
    (http://localhost:8069/web/static/src/js/compatibility.js:179:13)
    at OdooClass.WebClient.include.init
    (http://localhost:8069/web/static/src/js/compatibility.js:131:9)
    at OdooClass.init
    (http://localhost:8069/web/static/src/js/framework/class.js:122:38)
    at OdooClass.Class
    (http://localhost:8069/web/static/src/js/framework/class.js:106:33)
    at http://localhost:8069/web?debug:420:54
    at process_job (http://localhost:8069/web/static/src/js/boot.js:210:44)
    at Object.window.odoo.process_jobs
    (http://localhost:8069/web/static/src/js/boot.js:252:17)
    at Object.window.odoo.define
    (http://localhost:8069/web/static/src/js/boot.js:140:18)
```

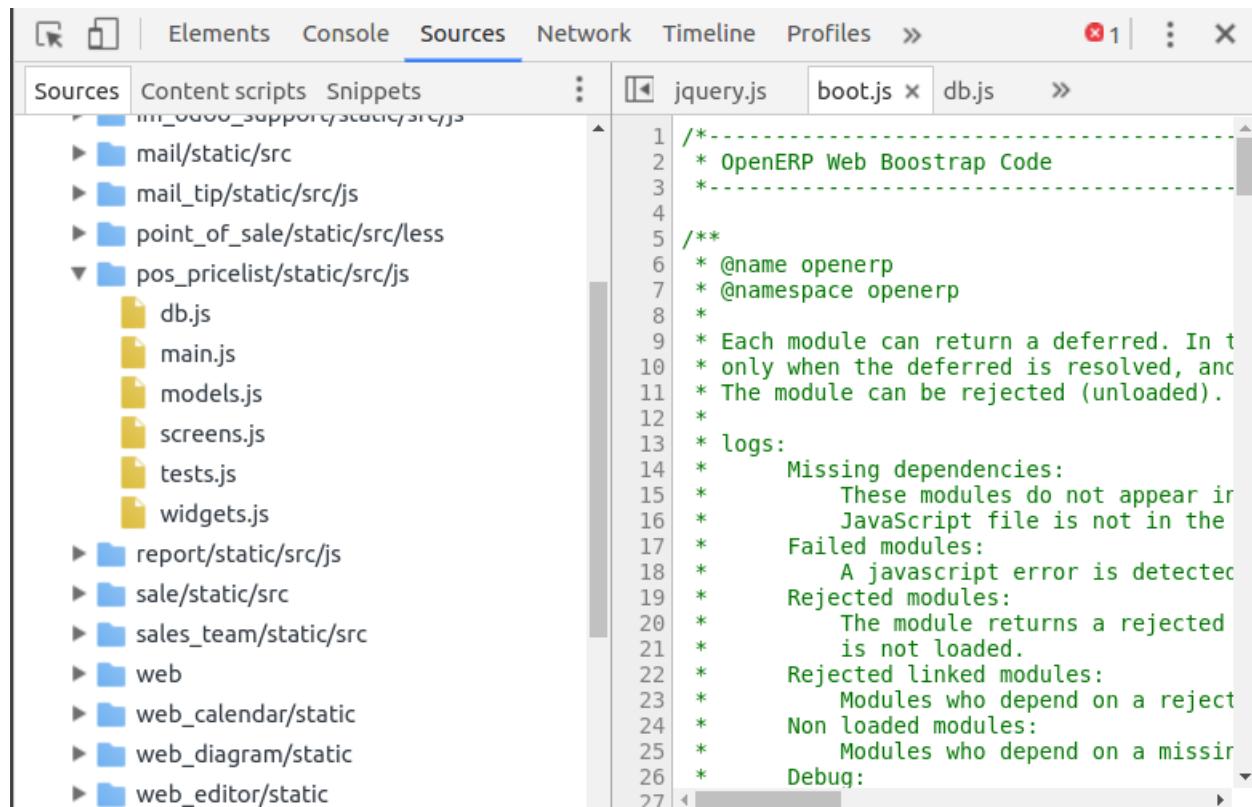
### 3.6.2 Error: Missing dependencies

For example, sometimes during page load displayed the error type:

Missing dependencies: [...] Non loaded modules: [...]



You can find out reason in the Developer Tool in the tab Sources as described above.

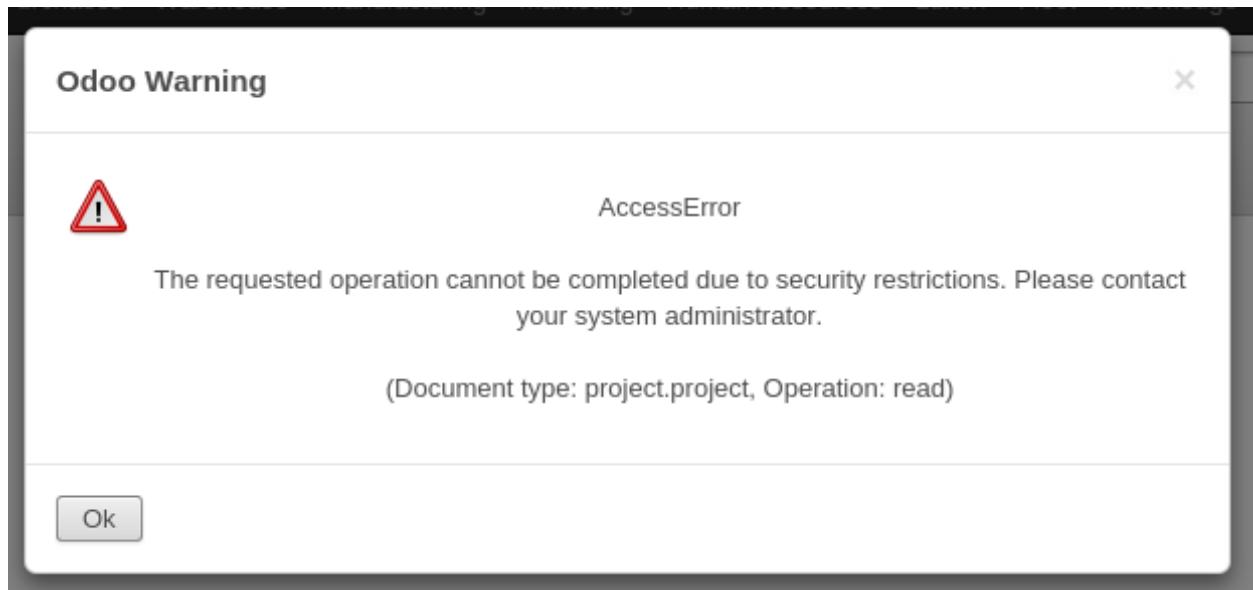


Likely you can not find files included in the Missing dependencies list. Then you need to check they are included in the view (.xml) files.

### 3.6.3 AccessError: Please contact your system administrator

There is an AccessError which doesn't specify groups that have access to an operation. It simply states:

The requested operation cannot be completed due to security restrictions. Please contact your system administrator.



Such error means, that your user doesn't satisfy access requirements specified in [\*ir.rule\*](#). See [\*Access section\*](#) for general understanding how odoo security works.



# CHAPTER 4

---

## Quality assurance

---

### 4.1 Test automation

#### 4.1.1 Python Autotests

To add tests you need:

- Create folder named **tests**
- Add `__init__.py` file
- Create a file whose name starts with **test\_** (put corresponding import to `__init__.py` file from the previous step)
- Add new Class based on one of `test classes`
- Add test methods whose names start with **test\_**

**Warning:** you shall NOT import `tests` in module folder, i.e. do NOT add `from . import tests` to main `__init__.py` file

Example:

```
from odoo.tests.common import TransactionCase

class TestMessage(TransactionCase):
    at_install = True
    post_install = True

    def test_count(self):
        expected_value = self.do_something()
        actual_value = self.get_value()
        self.assertEqual(expected_value, actual_value)
```

(continues on next page)

(continued from previous page)

```
def do_something(self):  
    ...
```

### Documentation:

#### How to run tests

Use following parameters when you start odoo:

- --test-enable
- -d \$DB\_CONTAINER
- -i \$MODULE
- --workers=0

#### js tests

To run tests with phantomjs tests you also need:

- [Install phantomjs or use dockers \(see below\)](#)
- use --db-filter=.\*

#### Docker users

You don't need to remove docker container to run test. You can run it in a separate container

- don't worry about name for new container – just use --rm arg
- No need to expose ports

So, to run tests with docker:

- use an odoo database which has required modules installed (otherwise it will test all dependencies too)
- OPTIONAL: stop main odoo container, but keep db container
- run new container, e.g.:

```
docker run --rm --link $DB_CONTAINER:db \  
-v /something/at/host:/something/at/container \  
itprojectsllc/install-odoo:$ODOO_BRANCH-dev \  
-- \  
--test-enable \  
--workers=0 \  
--stop-after-init \  
-d $DATABASE_NAME \  
-i $MODULE
```

## Odoo unittest

- *Test classes*
- *setUp and other methods*
- *Assert Methods*

### Test classes

From `odoo/tests/common.py`:

```
class BaseCase(unittest.TestCase):
    """
    Subclass of TestCase for common OpenERP-specific code.

    This class is abstract and expects self.registry, self.cr and self.uid to be
    initialized by subclasses.
    """

class TransactionCase(BaseCase):
    """
    TestCase in which each test method is run in its own transaction,
    and with its own cursor. The transaction is rolled back and the cursor
    is closed after each test.
    """

class SingleTransactionCase(BaseCase):
    """
    TestCase in which all test methods are run in the same transaction,
    the transaction is started with the first test method and rolled back at
    the end of the last.
    """

class SavepointCase(SingleTransactionCase):
    """
    Similar to :class:`SingleTransactionCase` in that all test methods
    are run in a single transaction *but* each test case is run inside a
    rollbacked savepoint (sub-transaction).

    Useful for test cases containing fast tests but with significant database
    setup common to all cases (complex in-db test data): :meth:`~.setUpClass`
    can be used to generate db test data once, then all test cases use the
    same data without influencing one another but without having to recreate
    the test data either.
    """

class HttpCase(TransactionCase):
    """
    Transactional HTTP TestCase with url_open and phantomjs helpers.
    """
```

### setUp and other methods

For more information see <https://docs.python.org/2.7/library/unittest.html#test-cases>

- `setUp()` – Method called to prepare the test fixture. This is called immediately before calling the test method.  
It's recommended to use in `TransactionCase` and `HttpCase` classes

- `setUpClass()` – A class method called before tests in an individual class run. `setUpClass` is called with the class as the only argument and must be decorated as a `classmethod()`. It's recommended to use in `SingleTransactionCase` and `SavepointCase` classes

```
@classmethod  
def setUpClass(cls):  
    ...
```

- `tearDown()`, `tearDownClass` – are called *after* test(s). Usually are not used in odoo tests

### Assert Methods

<https://docs.python.org/2.7/library/unittest.html#assert-methods>

#### at\_install, post\_install

By default, odoo runs test with paramaters:

```
at_install = True  
post_install = False
```

#### at\_install

- runs tests right after loading module's files. It runs only in demo mode.
- runs as if other not loaded yet modules are not installed at all
- runs before marking module as installed, which also leads to not loading module's qweb without `fixing it manually` (don't forget to use `special environment` in odoo before version 12) .

#### post\_install

- runs after installing all modules in current installation set
- runs after calling `registry.setup_models(cr)`
- runs after calling `model._register_hook(cr)`

### 4.1.2 JS Autotests

For automatic web tests odoo uses phantomjs.

#### How to write automatic js tests:

- Follow instruction for `python tests`
- If you have to make several steps in UI to test something:
  - Create `tour`
  - `Run tour via self.phantom_js()`
- If just one step is enough:
  - Run your js code via `self.phantom_js()`

**Documentation:****self.phantom\_js()**

From `odoo/tests/common.py`:

```
def phantom_js(self, url_path, code, ready="window", login=None, timeout=60, **kw):
    """ Test js code running in the browser
    - optionnally log as 'login'
    - load page given by url_path
    - wait for ready object to be available
    - eval(code) inside the page
    To signal success test do:
    console.log('ok')
    To signal failure do:
    console.log('error')
    If neither are done before timeout test fails.
    """

```

i.e.

- odoo first loads `url_path` as user login (e.g. '`admin`', '`demo`' etc.) or as non-authed user
- then waits for `ready` condition, i.e. when some js variable (e.g. `window`) become `truthy`
- then executes js code
- then wait for one of condition:
  - someone prints `console.log('ok')` – test passed
  - someone prints `console.log('error')` – test failed
  - `timeout` seconds are passed – test failed

**Example**

Example from `mail_sent`:

```
# -*- coding: utf-8 -*-
import odoo.tests

@odoo.tests.common.at_install(False)
@odoo.tests.common.post_install(True)
class TestUi(odoo.tests.HttpCase):

    def test_01_mail_sent(self):
        # wait till page loaded and then click and wait again
        code = """
            setTimeout(function () {
                $(".mail_sent").click();
                setTimeout(function () {console.log('ok');}, 3000);
            }, 1000);
        """
        link = '/web#action=%s' % self.ref('mail.mail_channel_action_client_chat')
        self.phantom_js(link, code, "odoo.__DEBUG__.services['mail_sent.sent'].is_
        ↪ready", login="demo")
```

In this test:

- odoo first loads /web#action=... page
- then waits for `odoo.__DEBUG__.services['mail_sent.sent'].is_ready`
  - `odoo.__DEBUG__.services['mail_sent.sent']` is similar to `require('mail_sent.sent')`
  - `is_ready` is a variable in `sent.js`
- then executes js code:

```
setTimeout(function () {
    $(".mail_sent").click();
    setTimeout(function () {console.log('ok');}, 3000);
}, 1000);
```

which clicks on Sent menu and gives to the page 3 seconds to load it.

This code neither throws errors (e.g. via `throw new Error('Some error description')`) nor log `console.log('error')`, but you can add ones to your code to catch failed cases you need.

- then if everything is ok, odoo get message `console.log('ok')`

### JS tests via Tours

How to run *odoos tours* in `phantom_js` method?

#### 10.0+

```
from odoo.tests.common import HttpCase

class CLASS_NAME(HttpCase):
    def test_NAME(self):

        tour = 'TOUR_NAME'
        self.phantom_js(
            URL_PATH,
            "odoo.__DEBUG__.services['web_tour.tour']"
            ".run('%s')" % tour,

            "odoo.__DEBUG__.services['web_tour.tour']"
            ".tours['%s'].ready" % tour,
            login=LOGIN_OR_NONE
        )
```

#### 8.0, 9.0

```
class CLASS_NAME(...):
    def test_NAME(self):

        self.phantom_js(
```

(continues on next page)

(continued from previous page)

```

URL_PATH,
    "odoo.__DEBUG__.services['web.Tour']"
    ".run('TOUR_NAME', 'test')",
    "odoo.__DEBUG__.services['web.Tour']"
    ".tours.TOUR_NAME",
    login=LOGIN_OR_NONE
)

```

## How js tour works via phantomjs

The order is as following:

- OPEN *url\_path* from **python phantom\_js** method
- WAIT *ready* condition (Truthy or Falsy) from **python phantom\_js** method
- OPEN *url* from *tour*'s options in **js** file
- WAIT *wait\_for* (deferred object) from *tour*'s options in **js** file
- DO first step from **js tour**
  - WAIT when *trigger* becomes visible
  - WAIT when *extra\_trigger* becomes visible (if *extra\_trigger* is presented)
  - EXECUTE action (*run* or click on *trigger*)
- DO NEXT step
  - ...
- STOP Running when:
  - error happens:
    - \* thrown via `raise`
    - \* reported via `console.log('error', ...)`
    - \* reported via `console.error(...)`, etc.
    - \* reported by tour system on **timeout** for initial *ready* condition. Timeout value is 10 sec and it cannot be changed. The PR to odoo 12 to make it customizable: <https://github.com/odoo/odoo/pull/24514>
    - \* reported by tour system on step **timeout**.
  - '`ok`' is reported via `console.log('ok')`
    - \* directly by code
    - \* indirectly by tour system when all steps are done
  - **timeout** from **python phantom\_js** method is occurred. Default is 60 sec

## Phantom\_js + python tests

### Odoo 12.0+

Since Odoo 12.0 there is no any problem with mixing calling phantom\_js and python code

### Odoo 11.0-

If you need you run some python code before or after calling phantom\_js you shall not use `self.env` and you need to create new env instead:

```
phantom_env = api.Environment(self.registry.test_cr, self.uid, {})
```

This is because `HttpCase` uses special cursor and using regular cursor via `self.env` leads to deadlocks or different values in database.

## Screenshots in PhantomJS tests

Open file `odoo/tests/phantomtest.js` and after the line

```
console.log("PhantomTest.run: execution launched, waiting for console.log('ok')...");
```

add following

```
i=1;
setInterval(function() {
    self.page.render('/tmp/phantomjs-'+i+'.png');
    i++;
}, 1000);
```

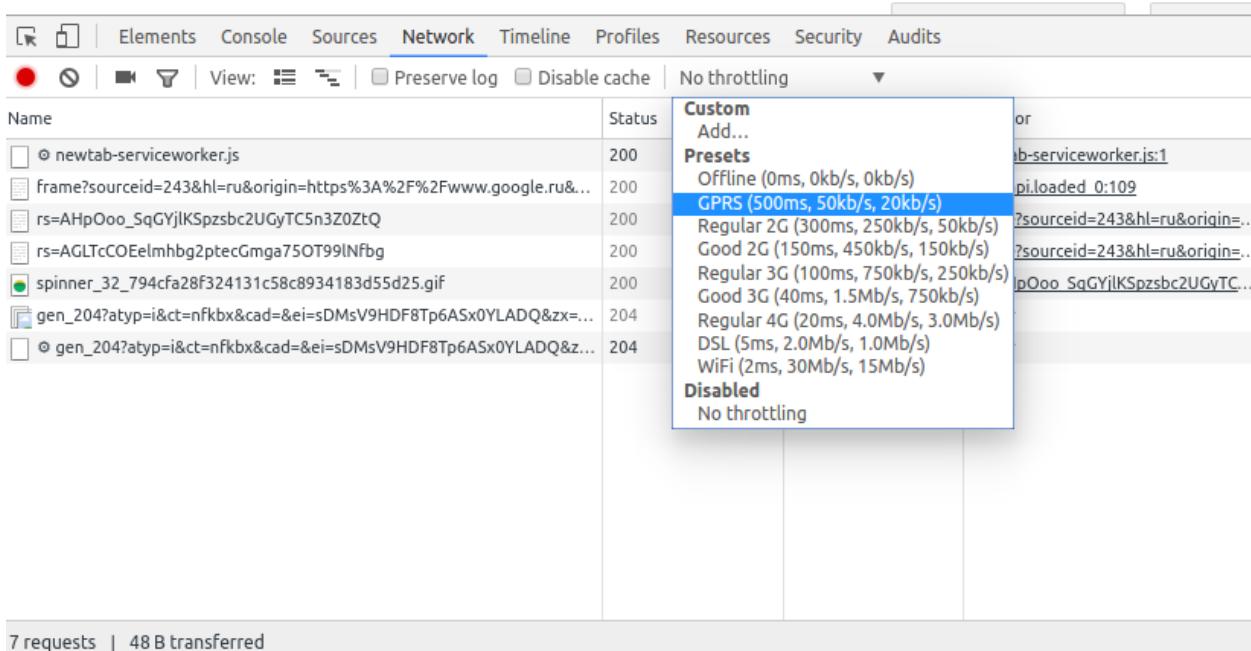
It will create screenshot every 1 second (you can update it if needed)

## Longpolling in unit tests

It's not possible.

## 4.2 Manual testing

### 4.2.1 Emulation of slow internet connections in browser



### Emulation of package lossing

In case if you need to emulate *bad* connection, i.e. it works and probably fast, but lose some percents of TCP packages, then do as following

```
# check your network interfaces
ifconfig

# Example below is for eth0
# Other possbile values are
# * wlan0 - wireless connection
# * lo - local connection. Use this, if your run a server on your machine

# lose 30 %
sudo tc qdisc add dev eth0 root netem loss 30%

# "burst of losing"
# Probabilty of each next lossing depends on previous result.
# For example below:
# Pnext = 0.1 * Pprev + (1-0.1) * Random(0,1)
# Then the package is lost, if Pnext < 0.3
sudo tc qdisc add dev eth0 root netem loss 30% 10%

# show current settings
tc -s qdisc show dev eth0

# reset settings
sudo tc qdisc del dev eth0 root
```

## 4.2.2 Emulation barcode

Barcode scanner connected with computer work as keyboard. E.g. after scanning send sequence of symbols as if fast typing on the keyboard.

Install **xdotool** app if you haven't it yet.

```
sudo apt-get install xdotool
```

Emulation scanning barcode:

```
sleep 3 && echo '1234567890128' | grep -o . | xargs xdotool key && xargs xdotool key ↵\n &
```

or so:

```
sleep 3 && echo '333333333338' | grep -o . | xargs xdotool key && xargs xdotool key ↵\n &
```

Where: 3 - sleep seconds; 333333333338 - barcode.

After successfully scanning you will see '333333333338' in the command line. If toggle to other window that symbols appear in the input field in the this window. So we can send sequence in the app as if we scanning it.

## 4.2.3 ESC/POS printer emulation

### hw\_escpos

- apply patch

```
cd /path/to/odoo/  
  
# odoo 10  
curl https://raw.githubusercontent.com/it-project-l1c/odoo-development/master/ ↵docs/debugging/hw_escpos-patch/hw_escpos-10.patch > hw_escpos.patch  
  
# odoo 9  
curl https://raw.githubusercontent.com/it-project-l1c/odoo-development/master/ ↵docs/debugging/hw_escpos-patch/hw_escpos-9.patch > hw_escpos.patch  
  
git apply hw_escpos.patch
```

- install hw\_escpos on odoo
- run a separate odoo with following args:

```
-d DB_WITH_HW_ESCPOS --db-filter=DB_WITH_HW_ESCPOS --xmlrpc-port=8888 --workers=0
```

- in new terminal run

```
tail -f /tmp/printer
```

On printing:

- some binary data is sent to /tmp/printer
- odoo prints logs with unparsed data

## POS

At any database (including one on runbot as well as database where you have installed hw\_escpos):

- set Receipt printer checkbox in pos.config and set ip equal to 127.0.0.1:8888
- open POS interface

**Warning:** for some reason printer emulation doesn't work in debug mode

- print ticket

### 4.2.4 Paypal testing

To test paypal payments you need to:

- Create developer account
- Add seller and buyer in developer sandbox
- Configure odoo
- Directly testing

#### Create developer account

Go to <https://developer.paypal.com/> and create new account.

#### Add seller and buyer

- Go to **Dashboard->Sand box->Accounts**. Create business (seller) and personal (buyer) accounts. It's recommended to don't use non-ascii symbols in account information (address, name etc.)
- Add some money to buyer (type amount in according field).
- Go to <http://sandbox.paypal.com> and login as seller. May be you will be forced to apply unconfirmed ssl certificate.
- Go to **Profile**.
- Copy *protected seller code*.

#### Configure odoo

- Install **payment\_paypal** module
- Go to **Settings->Payments->Payments->Paypal**.
- Pres **Edit**.
- Enter here **Paypal Email ID** - it is *seller* account.
- Enter **Paypal Merchant ID** - paste *protected seller code*.
- Set price list currency same as in paypal account.

### **Directly testing**

Open web shop. Buy some goods and pay with paypal. When you will be redirected on paypal page use *buyer* login and password.

# CHAPTER 5

---

## Porting Modules

---

### 5.1 Common Migration Helpers

- *Switching off unported modules*
- *Reviewing odoo updates*
- *Reviewing module source*

#### 5.1.1 Switching off unported modules

```
# (create fresh branch from upstream)

# mark all modules as non-installable
find . -type f -name __openerp__.py -or -name __manifest__.py | xargs sed -i 's/
˓→"installable": True/"installable": False/'
find . -type f -name __openerp__.py -or -name __manifest__.py | xargs sed -i "s/
˓→'installable': True/'installable': False/"

# check for fields without "installable" tag in manifest
find . -type f -name __openerp__.py -or -name __manifest__.py | xargs grep -L
˓→"installable.: "
# (if there is any output -- edit those files manually)

# prepare a commit
git add .
# check commit diff
git diff --cached
# Emoji prefixed with odoo version
git commit -m ":one::two::sos: mark unported modules as non-installable"
# (make "git push" and pull request at github)
```

## 5.1.2 Reviewing odoo updates

Code below helps you to find what is new between odoo branches

```
cd path/to/odoo/  
  
# check name for remote corresponding to https://github.com/odoo/odoo.git  
git remote -v  
  
# change directory to the module you need. To check core updates use "cd odoo/"  
cd addons/mail/  
  
git log \  
  --date=relative \  
  --pretty=format:"%h%x09%Cblue%ad%Creset%x09%ae%x09%Cgreen%s%Creset" \  
  --invert-grep \  
  --grep='\[FIX\]' \  
  --grep='\[MERGE\]' \  
  --grep='\[DOC\]' \  
  --grep='\[CLA\]' \  
  --grep='\[I18N\]' \  
origin/10.0..origin/11.0 -- . # use corresponding remote name and version
```

## 5.1.3 Reviewing module source

Commands below may help you to estimate amount of work to migrate module. The commands simply show all source in one view

```
# view source  
find . -iname "*.py" -or -iname "*.xml" -or -iname "*.csv" -or -iname "*.yml" -or -  
  -iname "*.js" -or -iname "*.rst" -or -iname "*.md" | xargs tail -n +1 | less  
  
# view source without docs  
find . -iname "*.py" -or -iname "*.xml" -or -iname "*.csv" -or -iname "*.yml" -or -  
  -iname "*.js" | xargs tail -n +1 | less
```

## 5.2 11.0- → 12.0+

### 5.2.1 New API

```
# renames in base modules  
find . -type f -name '*.py' | xargs sed -i 's/from odoo.addons.base.res/from odoo.  
  ↪addons.base.models/g'  
find . -type f -name '*.py' | xargs sed -i 's/from odoo.addons.base.ir/from odoo.  
  ↪addons.base.models/g'
```

## 5.3 10.0 → 11.0+, py2 → py3, shared Settings, sudo.get\_param()

### 5.3.1 New API

```
# ir.config_parameter -- prefix get_param / set_param with sudo()
find . -type f -name '*.py' | xargs perl -i -p0e 's/(?<!sudo\(\)\.).(get_param|set_
˓→param)/sudo().$1/g'
find . -type f -name '*.xml' | xargs perl -i -p0e 's/(?<!sudo\(\)\.).(get_param|set_
˓→param)/sudo().$1/g'

# page="True" is not used anymore
find . -type f -name '*.xml' | xargs sed -i 's/ page="True"///g'

# TODO: commands for "shared Settings" (python and xml)
```

### 5.3.2 New references

```
# mixins in js
find . -type f -name '*.js' | xargs sed -i 's/core\.mixins/require("web.mixins")/g'

# 11.0 doesn't have website.config.settings
find . -type f -name '*.py' -o -iname '*.xml' | xargs sed -i 's/website\.config\
˓→settings/res.config.settings/g'

# pos.config form
find . -type f -name '*.xml' | xargs sed -i 's/point_of_sale\.view_pos_config_form/
˓→point_of_sale\.pos_config_view_form/g'

# web.webclient_bootstrap template
find . -type f -name '*.xml' | xargs sed -i 's/web\.webclient_script/web\.webclient_
˓→bootstrap/g'

# 11.0 doesn't have base_action_rule module, it was was renamed to base_automation
find . -type f -name '*.xml' | xargs sed -i 's/base\.action\.rule/base\.automation/g'
find . -type f -name '*.py' | xargs sed -i "s/'base_action_rule'/'base_automation'/g"
find . -type f -name '*.py' | xargs sed -i 's/"base_action_rule"/"base_automation"/g'

# base.config.settings merged into res.config.settings
find . -type f -name '*.xml' -or -name "*.py" | xargs sed -i 's/base\.config\
˓→settings/res.config.settings/g'

# kanban_record in js
find . -type f -name '*.js' | xargs sed -i 's/web_kanban\.Record/web.KanbanRecord/g'

# Model -> rpc in require
find . -type f -name '*.js' | xargs sed -i "s/var Model = require('web\.Model');/var_
˓→rpc = require('web\.rpc');/g"
find . -type f -name '*.js' | xargs sed -i 's/var Model = require("web\.Model");/var_
˓→rpc = require("web\.rpc");/g'
```

### 5.3.3 Python 3

```
# coding: utf-8 is not needed anymore
find . -type f -name '*.py' | xargs sed -i '/# -\*- coding: utf-8 -\*-/d'

# urlparse
find . -type f -name '*.py' | xargs sed -i 's/import urlparse/import urllib.parse as_
˓→urlparse/g'
find . -type f -name '*.py' | xargs sed -i 's/from urlparse/from urllib.parse/g'
# StringIO
find . -type f -name '*.py' | xargs sed -i 's/from cStringIO import StringIO/from io_
˓→import StringIO/g'
find . -type f -name '*.py' | xargs sed -i 's/from StringIO import StringIO/from io_
˓→import StringIO/g'

# base64
# TODO
# SOMETHING.encode('base64') -> base64.b64encode(SOMETHING)
# SOMETHING.decode('base64') -> base64.b64decode(SOMETHING)
```

## 5.4 9.0- → 10.0+, \_\_openerp\_\_.py → \_\_manifest\_\_.py

### 5.4.1 New API

```
# rename all manifests
find . -type f -name __openerp__.py -exec rename 's/__openerp__.py/__manifest__.py/' '
˓→{}' \;
```

### 5.4.2 New references

```
# menu_hr_configuration
find . -type f -name '*.xml' | xargs sed -i 's/menu_hr_configuration/menu_human_
˓→resources_configuration/g'
# base.group_hr
find . -type f -name '*.csv' -o -name '*.py' -o -name '*.xml' | xargs sed -i 's/
˓→base.group_hr/hr.group_hr/g'
# website.salesteam_website_sales
find . -type f -name '*.csv' -o -name '*.py' -o -name '*.xml' | xargs sed -i 's/
˓→website.salesteam_website_sales/sales_team.salesteam_website_sales/g'
# base.group_sale_salesman
find . -type f -name '*.csv' -o -name '*.py' -o -name '*.xml' | xargs sed -i 's/
˓→base.group_sale_salesman/sales_team.group_sale_salesman/g'
# product.prod_config_main
find . -type f -name '*.xml' | xargs sed -i 's/product.prod_config_main/sale.prod_
˓→config_main/g'
```

## 5.5 7.0 → 8.0+, (cr, uid, ids, context) → self.env

### 5.5.1 Automatic replacements

```

# IMPORTS
# replace osv, orm
find . -type f -name '*.py' | xargs sed -i 's/from openerp.osv import orm$/from odoo_
˓→import models/g'
find . -type f -name '*.py' | xargs sed -i 's/from openerp.models.orm import Model$/
˓→from odoo.models import Model/g'
find . -type f -name '*.py' | xargs sed -i 's/osv.osv_memory/models.TransientModel/g'
find . -type f -name '*.py' | xargs sed -i 's/osv.osv/models.Model/g'
find . -type f -name '*.py' | xargs sed -i 's/osv.exception/UserError/g'
find . -type f -name '*.py' | xargs sed -i 's/osv./models./g'
find . -type f -name '*.py' | xargs sed -i 's/\\<orm./models./g'
find . -type f -name '*.py' | xargs sed -i 's/\\(import .*\\), osv/\\1, models/g'
find . -type f -name '*.py' | xargs sed -i 's/\\(import .*\\)osv, /\\1models, /g'
find . -type f -name '*.py' | xargs sed -i 's/\\(import .*\\)osv/\\1models/g'

find . -type f -name '*.py' | xargs sed -i 's/\\(import .*\\), orm/\\1/g'
find . -type f -name '*.py' | xargs sed -i 's/\\(import .*\\)orm, /\\1/g'
find . -type f -name '*.py' | xargs sed -i 's/^.*import orm$///g'

find . -type f -name '*.py' | xargs sed -i 's/openerp.osv/openerp/g'

# replace http import
find . -type f -name '*.py' | xargs sed -i 's/from openerp.addons.web import http/
˓→from odoo import http/g'
find . -type f -name '*.py' | xargs sed -i 's/openerp.addons.web.http/odoo.http/g'
find . -type f -name '*.py' | xargs sed -i 's/openerp.http/odoo.http/g'

# replace odoo
# fix importing. Otherwise you will get error:
#   AttributeError: 'module' object has no attribute 'session_dir'
find . -type f -name '*.py' | xargs sed -i 's/openerp.tools.config/odoo.tools.config/g
˓→'

# general replacement
find . -type f -name '*.py' | xargs sed -i 's/from openerp/from odoo/g'

# FIELDS
# update fields
# (multiline: http://stackoverflow.com/questions/1251999/how-can-i-replace-a-newline-
˓→n-using-sed/7697604#7697604 )
# delete _columns
find . -type f -name '*.py' | xargs perl -i -p0e 's/      _columns = {(.*)}\n      }/$1\n/
˓→gs'
# computed fields
find . -type f -name '*.py' | xargs sed -i 's/fields.function(\(.*)\)\n  ([\"\\x27][^,]*\\))/
˓→fields.function(\1 string=\2/g'
find . -type f -name '*.py' | xargs sed -i 's/fields.function(\(.*) multi=[^,])*/\n
˓→fields.function(\1/g'
find . -type f -name '*.py' | xargs sed -i 's/fields.function(\([^\,]*\\))\\(.*)type=.
˓→\\([2a-z]*\\)[\"\\x27]/fields.\3(compute="\1"\2/g'
find . -type f -name '*.py' | xargs sed -i 's/fields.many2one(\(.*)obj=\([^\,]*\\)/
˓→fields.many2one(\2, \1/g'

```

(continues on next page)

(continued from previous page)

```

find . -type f -name '*.py' | xargs sed -i 's/, [ ]*, //,/g'
find . -type f -name '*.py' | xargs sed -i 's/, [ ]*, //,/g'
find . -type f -name '*.py' | xargs sed -i 's/, [ ]*, //,/g'

# replace fields
find . -type f -name '*.py' | xargs perl -i -p0e 's/      _columns = {(.*)}    }/$1/gs'
find . -type f -name '*.py' | xargs sed -i 's/fields\.\.(.*)/fields.\u1/g'
find . -type f -name '*.py' | xargs sed -i 's/      [\x27"]\.(.*\.)[\x27"].*::.*\(fields.
˓→*`),$/\1 = \2/g'

# renamed attributes
find . -type f -name '*.py' | xargs sed -i 's/select=/index=/g'
find . -type f -name '*.py' | xargs sed -i 's/digits_compute=/digits=/g'

```

## 5.5.2 Semi-Automatic replacements

We recommend to use commands below after committing previous changes. It allows you to check differences.

The commands doesn't update code fully and usually you need to continue updates manually.

```

# pool -> env
find . -type f -name '*.py' | xargs sed -i 's/self.pool/self.env/g'
# remove cr, uid
find . -type f -name '*.py' | xargs sed -i 's/(cr, [^,]*, /(/g'
find . -type f -name '*.py' | xargs sed -i 's/(self, cr, [^,]*, ids/(self/g'
find . -type f -name '*.py' | xargs sed -i 's/(self, cr, uid, /(self, /g'
find . -type f -name '*.py' | xargs sed -i 's/, context=[^,)]*///g'
find . -type f -name '*.py' | xargs sed -i 's/self.env.get(\([^\)]*\))/self.env[\1]/g'
# res_config.py
find . -type f -name 'res_config.py' | xargs sed -i 's/\\(def get_default_.*\\)(self) /
˓→\\1(self, fields)/g'

```

**Warning:** Porting Modules is a process of adapting module to new version. E.g. we have module for odoo 10.0 and we want to make module work in odoo 11.0

As word *porting* is sometimes replaced to *migration*. You shall not confuse it with *Data Migration*, which sometimes is called just *migration*.

---

**Note:** We are happy to share our experience and hope that it will help someone to port odoo modules. We will be glad, if you share this page or recommend our team for module migration jobs:

- [it@it-projects.info](mailto:it@it-projects.info)
  - <https://www.it-projects.info/page/module-migration>
-

# CHAPTER 6

---

## User documentation

---

### 6.1 Module releasing checklist

This articles cover documentation and description part only.

#### 6.1.1 Module Name

- Module Name MUST be non-technical.

Examples of technical names:

- web\_debranding
- Web Debranding

Example of non-technical names:

- Backend Debranding

- Module Name MUST be the same at *manifest file*, *README.rst*, *doc/index.rst*

#### 6.1.2 Summary

- Review "summary" attribute at *manifest file* and first paragraph at *README.rst*. They MUST be presented, but MAY be different.

#### 6.1.3 Price

- Review "price" attribute at *manifest file*

## 6.1.4 Category

- Review "category" attribute at *manifest file*

## 6.1.5 doc/index.rst

- Review *content* and *formatting* of *doc/index.rst file*

## 6.1.6 README.rst

- Review *content* and *formatting* of *README.rst file*

## 6.1.7 static/description/index.html

- Prepare *HTML Description*
- Check *image sizes*

## 6.1.8 Main image

- Prepare image and specify it at "images" attribute at *manifest file*
- *Preview image at app store*

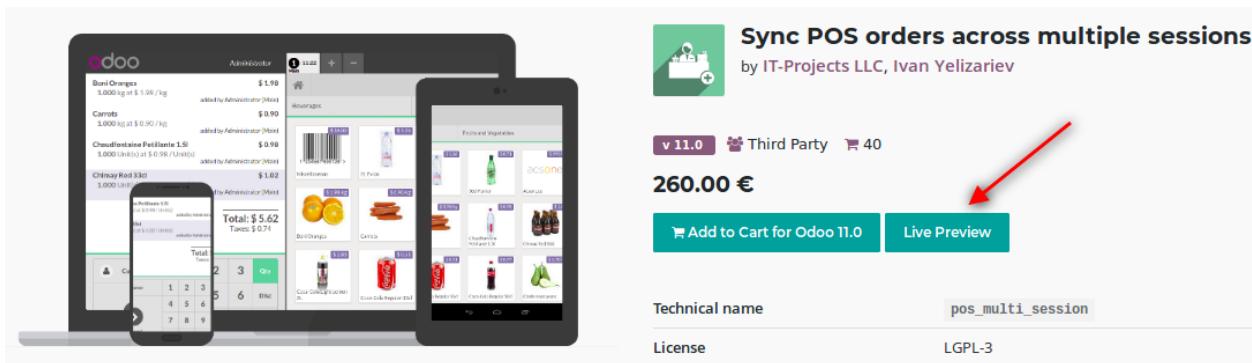
## 6.1.9 Live Preview

- Review "live\_test\_url" attribute at *manifest file*

For example:

```
"live_test_url": "http://apps.it-projects.info/shop/product/pos-multi-session?  
version=11.0",
```

- Live Preview button will appear at Odoo Apps Store after releasing the updates



## 6.2 static/description/index.html

- *Image sizes*
- *Templates*
  - *Title*
  - *Key features*
  - *Warnings and notes*
  - *Subsection*
  - *Reference to menu*
  - *Text + Image*
  - *Image + Text*
  - *Text, Image*
  - *Text, Image (large size)*
  - *Demo note*
  - *Contact us*
- *oe\_dark*

### 6.2.1 Image sizes

- *Image Sizes*

### 6.2.2 Templates

#### Title

```
<section class="oe_container">
    <div class="oe_row oe_spaced">
        <div class="oe_span12">
            <h2 class="oe_slogan" style="color:#875A7B;">NAME</h2>
            <h3 class="oe_slogan">SUMMARY OR SLOGAN</h3>
        </div>
    </div>
</section>
```

#### Key features

```
<section class="oe_container">
    <div class="oe_row oe_spaced">
        <div class="oe_span12">
            <div class="alert alert-info oe_mt32" style="padding:0.3em 0.6em; font-size: 150%;"
                <!-->
```

(continues on next page)

(continued from previous page)

```
<i class="fa fa-hand-o-right"></i><b> Key features: </b>
<ul class="list-unstyled">

    <li>
        <i class="fa fa-check-square-o text-primary"></i>
        FEATURE 1
    </li>

    <li>
        <i class="fa fa-check-square-o text-primary"></i>
        FEATURE 2
    </li>

</ul>
</div>

</div>
</div>
</section>
```

### Warnings and notes

Green:

```
<section class="oe_container">
    <div class="oe_row oe_spaced">
        <div class="oe_span12">

            <div class="alert alert-success oe_mt32" style="padding:0.3em 0.6em; font-size:_>
                ↪150%;">
                YOUR TEXT HERE
            </div>

        </div>
    </div>
</section>
```

Yellow:

```
<section class="oe_container">
    <div class="oe_row oe_spaced">
        <div class="oe_span12">

            <div class="alert alert-warning oe_mt32" style="padding:0.3em 0.6em; font-size:_>
                ↪150%;">
                YOUR TEXT HERE
            </div>

        </div>
    </div>
</section>
```

Red:

```
<section class="oe_container">
    <div class="oe_row oe_spaced">
        <div class="oe_span12">

            <div class="alert alert-danger oe_mt32" style="padding:0.3em 0.6em; font-size: 150
→%; ">
                YOUR TEXT HERE
            </div>

        </div>
    </div>
</section>
```

## Subsection

```
<h4 class="oe_slogan"><b>SUBSECTION NAME</b></h4>
```

(Put it inside `<section class="... "><div class="oe_row oe_spaced"> tags`)

## Reference to menu

To specify references to menu, use right arrow character `&rarr;`, for example:

```
Go to <em>Sales &rarr; Configuration &rarr; Settings</em>
```

## Text + Image

```
<section class="oe_container oe_dark">
    <div class="oe_row oe_spaced">
        <div class="oe_span6">
            <p class="oe_mt32">
                TEXT
            </p>
        </div>
        <div class="oe_span6">
            <div class="oe_row_img oe_centered" align="center">
                
            </div>
        </div>
    </div>
</section>
```

## Image + Text

```
<section class="oe_container oe_dark">
    <div class="oe_row oe_spaced">
        <div class="oe_span6">
            <div class="oe_row_img oe_centered" align="center">
                
            </div>
        </div>
    </div>
```

(continues on next page)

(continued from previous page)

```
</div>
<div class="oe_span6">
    <p class="oe_mt32">
        TEXT
    </p>
</div>
</div>
</section>
```

## Text, Image

```
<section class="oe_container oe_dark">
    <div class="oe_row oe_spaced">
        <div class="oe_span12 text-center">
            <p class="oe_mt32">
                TEXT
            </p>
        </div>
        <div class="oe_row_img oe_centered">
            
        </div>
    </div>
</section>
```

## Text, Image (large size)

```
<section class="oe_container oe_dark">
    <div class="oe_row oe_spaced">
        <div class="oe_span12 text-center">
            <p class="oe_mt32">
                <font style="font-size: 120%;">TEXT</font>
            </p>
        </div>
        <div class="oe_screenshot" align="center">
            
        </div>
    </div>
</section>
```

## Demo note

```
<section class="oe_container">
    <div class="oe_row oe_spaced">
        <div class="oe_span8">
            <h2>Want to take a look?</h2>
            <p class="oe_mt32">For a live demonstration click <em>LIVE PREVIEW</em>  

            ↵button above (near to <em><i class="fa fa-shopping-cart"></i> Add to Cart</em>) </p>
        </div>
    </div>
</section>
```

## Contact us

- *Contact us block*

### 6.2.3 oe\_dark

Use oe\_dark class on every even section. Don't use oe\_dark for beginning and ending sections.

```
<section class="oe_container">
    <!--Title-->
</section>

<section class="oe_container">
    <!--Key features-->
</section>

<section class="oe_container">
</section>

<section class="oe_container oe_dark">
</section>

<section class="oe_container">
</section>

<section class="oe_container oe_dark">
</section>

<section class="oe_container">
</section>

<section class="oe_container">
    <!--Free support section-->
</section>

<section class="oe_container">
    <!--Contact us block-->
</section>
```

## 6.3 Screenshots tools

- Nimbus Screen Screenshot: <http://nimbus.everhelper.me/screenshot.php>
- Shutter: <http://shutter-project.org/>

```
sudo add-apt-repository ppa:shutter/ppa
sudo apt-get update && sudo apt-get install shutter
```

## 6.4 Module description

### 6.4.1 Banner

The Banner is displayed only in Odoo Apps. It should be located in the `path_to_module/images/` directory and its size should not exceed 1500x1000 px. Next, in the `__openerp__.py` file you need make the relevant record:

```
"images": ["images/banner.png"],
```

### 6.4.2 Icon and index.html

The module icon needs to be located at `path_to_module/static/description/` and it must be called `icon.png`. Also in this directory you need to create `index.html`, where will be contained necessary HTML tags, text description and screenshots (the recommended size is 752x352 px).

#### See also:

See the official template <https://github.com/odoo/odoo/blob/master/addons/crm/static/description/index.html>

It is important that `index.html` and screenshots it contains should be included at the same folder.

The result of the `index.html` and icon appearance can be checked by opening the module in “Local Modules” of your Odoo instance.

### 6.4.3 Summary

This is an overview of content that provides a reader with the overarching theme, but does not expand on specific details.

Summary should be included at `__openerp__.py` as `'summary': """Summary text"""`. For example:

```
'summary': """Use multiple POS for handling orders"""
```

## 6.5 Contact us block

For every selling modules IT-Projects LLC adds block generated by following command:

```
export ODOO_BRANCH=11.0
echo && echo && \
curl --silent https://raw.githubusercontent.com/it-projects-llc/odoo-development/ \
master/docs/description/contactus.html \
| sed "s/ODOO_BRANCH/$ODOO_BRANCH/g" \
| sed "s/STAMP1_ROTATION/$($RANDOM % 20 - 10)/g" \
| sed "s/STAMP2_ROTATION/$($RANDOM % 20 - 10)/g" && \
echo && echo
```

## 6.6 JS Tour

Tour is a set of steps of possible scenario of module usage.

Steps may be executed automatically for `testing` purpose or by user for `demonstrating` purpose.

- *Tour Definition*
  - 10.0+
    - \* *Example*
    - \* *Options*
    - \* *Step*
    - \* *Predefined steps*
    - \* *More documentation*
  - 8.0, 9.0
    - \* *Example*
    - \* *Tour.register*
    - \* *Step*
    - \* *More documentation*
- *Open backend menu*
  - 11.0+
  - 9.0, 10.0
    - \* *Manifest*
    - \* *load\_xmlid*
    - \* *Tour*
  - 8.0
- *Manual launching*
  - 10.0+
  - 8.0, 9.0
- *Auto Launch after installation*
  - 10.0+
  - 8.0, 9.0

### 6.6.1 Tour Definition

**10.0+**

#### Example

Example from `website_sale` module:

```
odoo.define('website_sale.tour', function (require) {
'use strict';

var tour = require("web_tour.tour");
```

(continues on next page)

(continued from previous page)

```

var base = require("web_editor.base");

var options = {
    test: true,
    url: '/shop',
    wait_for: base.ready()
};

var tour_name = 'shop_buy_product';
tour.register(tour_name, options,
    [
        {
            content: "search ipod",
            trigger: 'form input[name="search"]',
            run: "text ipod",
        },
        {
            content: "search ipod",
            trigger: 'form:has(input[name="search"]) .oe_search_button',
        },
        {
            content: "select ipod",
            trigger: '.oe_product_cart a:contains("iPod")',
        },
        {
            content: "select ipod 32GB",
            extra_trigger: '#product_detail',
            trigger: 'label:contains(32 GB) input',
        },
        {
            content: "click on add to cart",
            extra_trigger: 'label:contains(32 GB) input:propChecked',
            trigger: '#product_detail form[action^="/shop/cart/update"] .btn',
        },
        /* ... */
    ]
);
}) ;
});

```

## Options

Options (second argument of `tour.register`):

- **test** – only for tests
- **url** – open link before running the tour
- **wait\_for** – wait for deferred object before running the script
- **skip\_enabled** – adds *Skip* button in tips

## Step

Each step may have following attributes:

- **content** – name or title of the step
- **trigger** (mandatory) – where to place tip. *In js tests: where to click*
- **extra\_trigger** – when this becomes visible, the tip is appeared. *In js tests: when to click*
- **timeout** – max time to wait for conditions
- **position** – how to show tip (left, right, top, bottom), default right
- **width** – width in px of the tip when opened, default 270
- **edition** – specify to execute in “community” or in “enterprise” only. By default empty – execute at any edition.
- **run** – what to do when tour runs automatically (e.g. in tests)
  - 'text SOMETEXT' – writes value in **trigger** element
  - 'click'
  - 'drag\_and\_drop TO\_SELECTOR'
  - 'auto' – auto action (click or text)
  - function: (actions) { ... } – actions is instance of RunningTourActionHelper – see [tour\\_manager.js](#) for its methods.
- **auto** – step is skipped in non-auto running

## Predefined steps

- tour.STEPS.MENU\_MORE – clicks on menu *More* in backend when visible
- tour.STEPS.TOGGLE\_APPSWITCHER – navigate to Apps page when running in enterprise
- tour.STEPS.WEBSITE\_NEW\_PAGE – clicks create new page button in frontend

## More documentation

- <https://www.odoo.com/slides/slides/the-new-way-to-develop-automated-tests-beautiful-tours-440>
- [https://github.com/odoo/odoo/blob/10.0/addons/web\\_tour/static/src/js/tour\\_manager.js](https://github.com/odoo/odoo/blob/10.0/addons/web_tour/static/src/js/tour_manager.js)
- [https://github.com/odoo/odoo/blob/10.0/addons/web\\_tour/static/src/js/tip.js](https://github.com/odoo/odoo/blob/10.0/addons/web_tour/static/src/js/tip.js)

## 8.0, 9.0

### Example

```
{
  id: 'mails_count_tour',
  name: _t("Mails count Tour"),
  mode: 'test',
  path: '/web#id=3&model=res.partner',
  steps: [
    {
      title: _t("Mails count tutorial"),
      content: _t("Let's see how mails count work."),
      popover: { next: _t("Start Tutorial"), end: _t("Skip") },
    }
  ]
}
```

(continues on next page)

(continued from previous page)

```

},
{
    title:     _t("New fields"),
    content:   _t("Here is new fields with mails counters. Press one of it."),
    element:   '.mails_to',
},
{
    waitNot:   '.mails_to:visible',
    title:     _t("Send message from here"),
    placement: 'left',
    content:   _t("Now you can see corresponding mails. You can send mail to this partner right from here. Press <em>'Send a mesage'</em>."),
    element:   '.oe_mail_wall .oe_msg.oe_msg_composer_compact>div>.oe_compose_post',
},
]
}

```

## Tour.register

In odoo 8 tour defines this way:

```
(function () {
'use strict';
var _t = openerp._t;
openerp.Tour.register({ ...
```

In odoo 9 tour defines that way:

```
odoo.define('account.tour_bank_statement_reconciliation', function(require) {
'use strict';
var core = require('web.core');
var Tour = require('web.Tour');
var _t = core._t;
Tour.register({ ...
```

Important details:

- **id** - need to call this tour
- **path** - from this path tour will be started in test mode

## Step

Next step occurs when **all** conditions are satisfied and popup window will appear near (chose position in *placement*) element specified in *element*. Element must contain css selector of corresponding node. Conditions may be:

- **waitFor** - this step will not start if *waitFor* node absent.
- **waitNot** - this step will not start if *waitNot* node exists.
- **wait** - just wait some amount of milliseconds before **next** step.
- **element** - similar to *waitFor*, but *element* must be visible
- **closed window** - if popup window have close button it must be closed before next step.

Opened popup window (from previous step) will close automatically and new window (next step) will be shown.

Inject JS Tour file on page:

```
<template id="res_partner_mails_count_assets_backend" name="res_partner_mails_count_
↳assets_backend" inherit_id="web.assets_backend">
    <xpath expr=". " position="inside">
        <script src="/res_partner_mails_count/static/src/js/res_partner_mails_count_
↳tour.js" type="text/javascript"></script>
    </xpath>
</template>
```

## More documentation

Some docs is here (begin from 10 slide): <http://www.slideshare.net/openobject/how-to-develop-automated-tests> Also checkout here: <https://github.com/odoo/odoo/blob/9.0/addons/web/static/src/js/tour.js>

## 6.6.2 Open backend menu

### 11.0+

No additional actions are required.

### 9.0, 10.0

Some additional actions are required to work with backend menus in tours

## Manifest

Add `web_tour` to dependencies

```
"depends": [
    "web_tour",
],
# ...
"demo": [
    "views/assets_demo.xml",
    "views/tour_views.xml",
],
```

## load\_xmlid

You need to set `load_xmlid` for *each* menu you need to open. Recommended name for the file is `tour_views.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<odoo>
    <!-- Make the xmlid of menus required by the tour available in webclient -->
    <record id="base.menu_administration" model="ir.ui.menu">
        <field name="load_xmlid" eval="True"/>
```

(continues on next page)

(continued from previous page)

```
</record>
</odoo>
```

### Tour

Use *trigger* selector for both editions:

```
{
    trigger: '.o_app[data-menu-xmlid="base.menu_administration"], .oe_menu_
    ↪toggler[data-menu-xmlid="base.menu_administration"]',
    content: _t("Configuration options are available in the Settings app."),
    position: "bottom"
}
```

### 8.0

The only way to open menu is search by string, for example

```
{
    title:      "go to accounting",
    element:    '.oe_menu_toggler:contains("Accounting"):visible',
},
```

### 6.6.3 Manual launching

#### 10.0+

- *activate developer mode.*
- Click *Bug* icon (between *chat icon* and *Username* at top right-hand corner)
  - click *Start tour*
- Click *Play* button – it starts tour in auto mode

To run *test-only* tours (or to run tours in auto mode but with some delay) do as following:

- open browser console (F12 in Chrome)
- Type in console:

```
odoo.__DEBUG__.services['web_tour.tour'].run('TOUR_NAME', 1000); // 1000 is delay
↪in ms before auto action
```

#### 8.0, 9.0

You can launch tour by url of following format:

/web#/tutorial.mails\_count\_tour=true

where *mails\_count\_tour* is id of your tour.

## 6.6.4 Auto Launch after installation

**10.0+**

TODO

**8.0, 9.0**

To run tour after module installation do next steps.

- Create *ToDo*
- Create *Action*

ToDo is some queued web actions that may call *Action* like this:

```
<record id="base.open_menu" model="ir.actions.todo">
    <field name="action_id" ref="action_website_tutorial"/>
    <field name="state">open</field>
</record>
```

Action is like this:

```
<record id="res_partner_mails_count_tutorial" model="ir.actions.act_url">
    <field name="name">res_partner_mails_count Tutorial</field>
    <field name="url">/web#id=3&model=res.partner&#tutorial_extra.mails_
    ↵count_tour=true</field>
    <field name="target">self</field>
</record>
```

Here `tutorial_extra.mails_count_tour` is tour id.

Use eval to compute some python code if needed:

```
<field name="url" eval="'/web?debug=1&res_partner_mails_count=tutorial#id=
    ↵'+str(ref('base.partner_root'))+'&view_type=form&model=res.partner&#tutorial_extra.mails_count_tour=true'"/>
```

## 6.7 Preview module on App Store

Browser's dev tools allows to preview Module in App Store before actual uploading.

- open <https://www.odoo.com/apps>
- click Inspect Element on some application
- change text and images
- Done! Now can decide do you need make changes or keep current images and text

### 6.7.1 Preview image

While it's easy to change text, it's not obvious how to preview image.

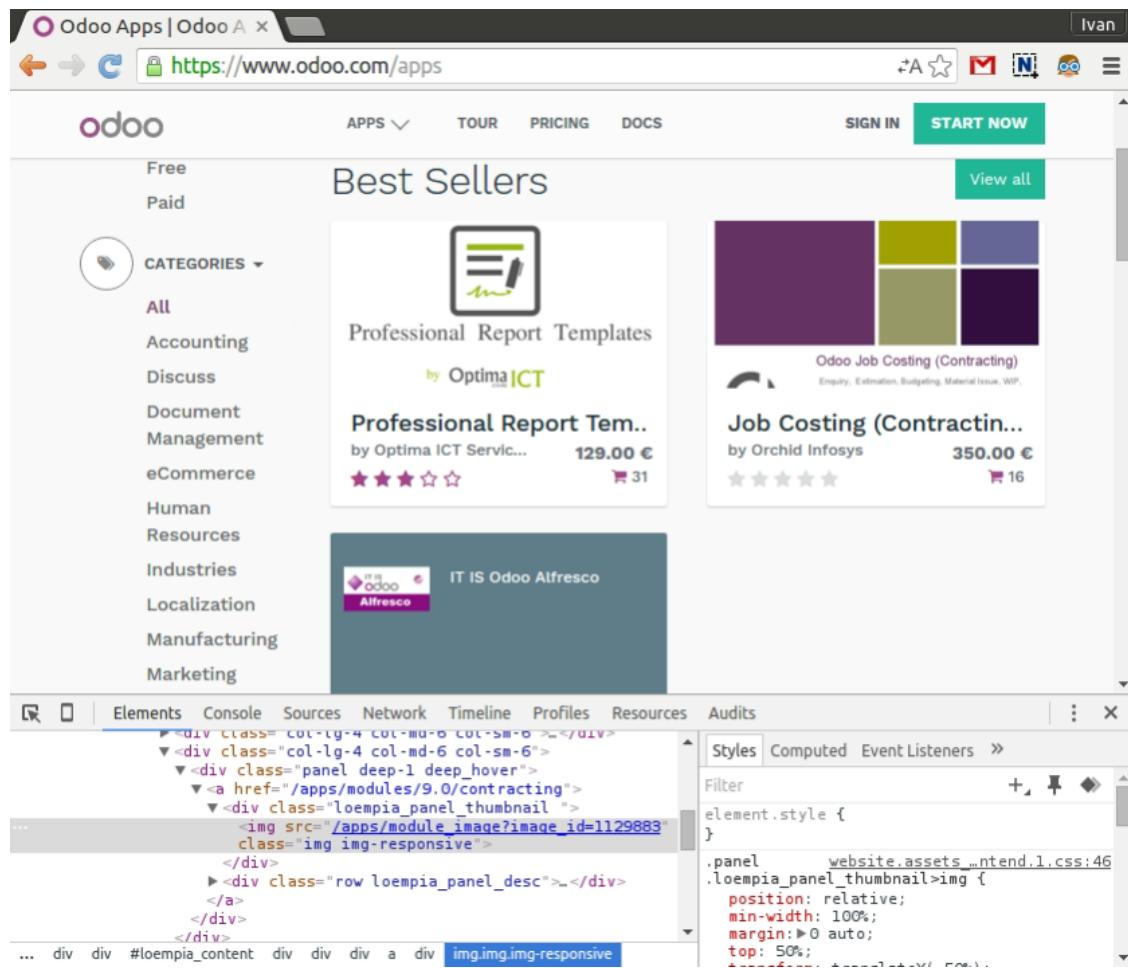
## Base64

- google: convert image to base64
- convert image to base64 with a tool you choosed. It must be some long string started with data:image/:

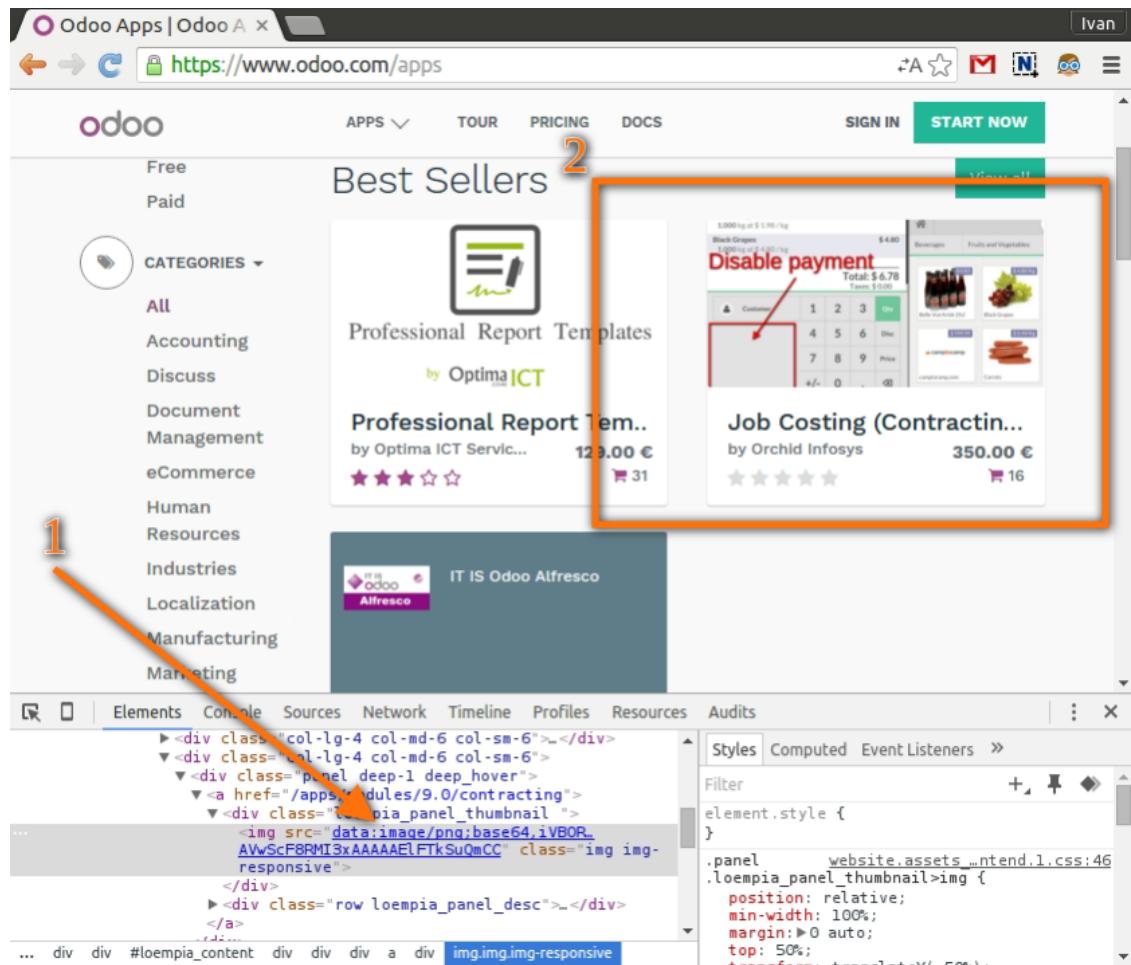
```
data:image/png;base64,iVBORw0KGgoAAAQAcF8RMI3xAAA.....AAE1FTkSuQmCC
```

- paste this line to src attribute of image tag

– BEFORE



– AFTER



## Nginx

Configure your nginx and use local link in src attribute.:

```

```

You cannot use localhost due to security restrictions. So, you need to add some domain to /etc/hosts::

```
127.0.0.1 static.local
```

TODO instruction for nginx

## 6.8 Image sizes

See also

- [Preview module on App Store](#)
- [Adjust chromium window size script](#)

- `__openerp__.py` -> ‘images’
- `description/index.html`

### 6.8.1 `__openerp__.py` -> ‘images’

This images is displayed on application page ([example](#)) and in application list ([example](#))

Displayed size:

- app page:

750 x 400

- app list:

262,5 x 130

Recommended size (aspect) to fit both usage:

750 x 371

You can scale picture, saving proportion.

---

**Note:** Appearance in *app list* is more important, as there is less chance that user open *app page*, if small sized image in *app list* is not attractive enough.

---

### 6.8.2 `description/index.html`

All values assumed, that you put the code inside `.oe_container` and `.oe_row`, e.g.:

```
<section class="oe_container">
    <div class="oe_row oe_spaced">
        ...
        <div class="oe_demo oe_picture oe_screenshot">
            
        </div>
        ...
    </div>
</section>
```

#### `oe_span6 img.oe_demo.oe_picture.oe_screenshot`

:: max-width: 362px; max-height: 382px;

#### `img.oe_demo.oe_picture.oe_screenshot`

:: max-width: 761px; max-height: 382px;

**img.oe\_demo.oe\_screenshot**

:: max-width: 928px;

**img.oe\_screenshot**

:: max-width: 1500px; max-height: 1000px;



## Git and Github

---

### 7.1 Initial git & github configuration

#### 7.1.1 ssh keys

Configure github ssh keys: <https://help.github.com/articles/connecting-to-github-with-ssh/>

#### 7.1.2 gpg keys

- Generate gpg keys: <https://help.github.com/articles/generating-a-new-gpg-key/>
- Add gpg key to github: <https://help.github.com/articles/adding-a-new-gpg-key-to-your-github-account/>
- Tell to git which key to use <https://help.github.com/articles/telling-git-about-your-gpg-key/>
- Tell git to sign all commits:

```
git config --global commit.gpgsign true
```

- Make gpg remember your passphrase

```
# Update gpg-agent config
# 28800 is 8 hours
echo "default-cache-ttl 28800" >> ~/.gnupg/gpg-agent.conf
echo "max-cache-ttl 28800" >> ~/.gnupg/gpg-agent.conf

# tell git to use gpg-agent
git config --global gpg.program gpg2

# install gpg2 if needed
sudo apt-get install gnupg2

# restart gpg-agent
```

(continues on next page)

(continued from previous page)

```
gpgconf --kill gpg-agent  
gpg-agent --daemon
```

- Make a backup if needed

```
# make backup file and move it to secret place  
gpg --export-secret-keys > secret-backup.gpg  
  
# you will be able to restore keys by following command:  
gpg --import secret-backup.gpg  
# or  
gpg2 --import secret-backup.gpg
```

**Warning:** If you lost your key or forgot password, you need to create new one, but don't remove old one from github, because otherwise all signed by old key commits will become "Unverified"

### 7.1.3 git email

- Configure email in `git`. Email must be the same as in github settings:

```
git config --global user.email "your_email@example.com"
```

### 7.1.4 git editor

```
git config --global core.editor "nano"
```

### 7.1.5 gitignore

- Configure global gitignore

Possible content for `~/.gitignore_global`:

```
*~  
*.pyc
```

## 7.2 Porting

If you add some feature to one branch and need to add it to another branch, then you have to make *port*.

See also:

- *Conflicts resolving*

### 7.2.1 Forward-port

It's the simplest case. You merge commits from older branch (e.g. 8.0) to newer branch (e.g. 9.0)

```
git checkout 9.0
git merge origin/8.0

# [Resolve conflicts if needed]

git push
```

After `git merge` you probably need to make some minor changes. In that case just add new commits to newer branch

```
git add ...
git commit -m "...."
git push
```

## 7.2.2 Back-port

If you need to port new feature from newer branch (e.g. 9.0) to older one (e.g. 8.0), then you have to make *back-port*.

The problem here is that newer branch has commits which should be applied for newer branch only. That is you cannot just make `git merge 9.0`, because it brings 9.0-only commits to 8.0 branch. Possible solutions here are:

## 7.2.3 git cherry-pick

Apply commits from newer branch (e.g. 9.0) to older branch (e.g. 8.0)

```
git checkout 8.0

git cherry-pick <commit-1>
# [Resolve conflicts if needed]

git cherry-pick <commit-2>
# [Resolve conflicts if needed]
# ...

git push
```

Also possible to pick the commit from any remote repository. Add this repository to your remotes. Do fetch from it. And then cherry-pick.

### cherry-pick range of commits

The command `git cherry-pick A..B` applies commits between A and B, but without A (A must be older than B). To apply inclusive range of commits use format as follows:

```
git cherry-pick A^..B
```

For example, to backport this PR <https://github.com/itProjects-llc/odoo-saas-tools/pull/286/commits>, use command:

```
git cherry-pick 6ee4fa07d4c0adc837d7061e09da14638d8abf8d^..
˓→ 9133939a25f9e163f52e6662045fc2dc6010ac14
```

## 7.3 Conflict resolving

After making `git merge` or `git cherry-pick` there could be conflicts, because some commits try to make changes on the same line. So, you need to choose which change shall be used. It could be one variant, both variants or new variant.

What to do if you got conflicts:

- Check status

```
git status
```

- Resolve conflicts:

- either edit files manually:

- \* open file with conflicts
    - \* search for <<< or >>> and delete obsolete variant or make a mix of both variants.

- or use following commands, if you are sure which version should be kept

```
git checkout --ours -- <file>
# or
git checkout --theirs -- <file>
```

- Mark files as resolved via `git add` command
- Done.

```
git push
```

### 7.3.1 Deleted files

Sometimes, changes can be conflicted because files are not exist anymore in *ours* version, but updated in *theirs* (or vice versa). In that case execute the code below in order to ignore such changes:

```
git status | grep 'deleted by us' | awk '{print $4}' | xargs git rm
git status | grep 'deleted by them' | awk '{print $4}' | xargs git rm
```

### 7.3.2 Notes

- It's important, that on resolving conflict stage you should not make any updates inside conflicting lines. You can only choose which lines should be kept and which deleted. E.g. if you resolve conflicts due to porting some updatefeature from one odoo version (e.g. 8.0) to another (e.g. 9.0), then such changes some time must be tuned to make updatefeature work on target odoo version. But you have to make such tuning on a new commit only. Make merging/cherry-picking commits be only about merging and cherry-picking, make porting commits separately.
- If you don't have conflicts, you do not need to make commit after cherry-pick because it creates commit by its own.

## 7.4 Multi Pull Request

### 7.4.1 Find last merged point

To find last commit upstream/8.0 and upstream/9.0 were merged, use following commands

```
git fetch
git log upstream/8.0..upstream/9.0 --grep="Merge remote-tracking branch 'origin/8.0'" ↵
↪--merges -n 3

# you will get something like that:
# commit 5cb3652be72a05330c3988d270f3aef548511b29
# Merge: f1cd564 6cc2562
# Author: Ivan Yelizariev <yelizariev@it-project.info>
# Date:   Sat Feb 27 16:00:42 2016 +0500
#
#       Merge remote-tracking branch 'origin/8.0' into 9.0-dev
#
# commit 14632a790aa01ee2a1ee9fe52152cf2fbfa86423
# Merge: 7a48b3a d66ba4f
# Author: Ivan Yelizariev <yelizariev@it-project.info>
# Date:   Thu Feb 25 11:31:43 2016 +0500
#
#       Merge remote-tracking branch 'origin/8.0' into 9.0-dev
#
# commit 6981c245afdcc39b2b49585f8205a784161f9c6
# Merge: 22081ed 6eb9f8d
# Author: Ivan Yelizariev <yelizariev@it-project.info>
# Date:   Fri Feb 19 19:14:15 2016 +0500
#
#       Merge remote-tracking branch 'origin/8.0' into 9.0-dev

# take one commit sha from the list and check that it's in origin/9.0.

git branch -r --contains 5cb3652be72a05330c3988d270f3aef548511b29

# possible output:
# upstream/9.0
# origin/9.0-dev

# if there is not upstream/9.0 in output,
# then commit has not been merged yet and you cannot use it
# for branch 9.0 use this commit sha 5cb3652be72a05330c3988d270f3aef548511b29
# for branch 8.0 need find which of two commits in ``Merge:`` line contains "upstream/
↪8.0"

git branch -r --contains f1cd564
git branch -r --contains 6cc2562

# Use commit sha to create new branches:

git checkout -b '9.0-new_branch_name' 5cb3652be72a05330c3988d270f3aef548511b29
git checkout -b '8.0-new_branch_name' 6cc2562
```

## 7.5 Cancel lame commit

Imagine you make lame commit. Now to repair things do next:

1. git reset HEAD~1 --soft
2. git status

You will see: Your branch is behind ‘origin/8.0’ by 1 commit, and can be fast-forwarded. (use “git pull” to update your local branch)

3. git add // Add here changed (fixed) files
4. git diff --cached //make sure everything is ok.
5. git status

You will see: Your branch is behind ‘origin/8.0’ by 1 commit, and can be fast-forwarded. (use “git pull” to update your local branch)

6. git commit -m'I fixed my mistakes'
7. git status

You will see: Your branch and ‘origin/8.0’ have diverged, and have 1 and 1 different commit each, respectively. (use “git pull” to merge the remote branch into yours)

Now finaly force is with you:

8. git push origin 8.0 -f

## 7.6 Pull request from console

Yes it possible! Try this manual: <https://github.com/github/hub> Than in console:

```
alias git=hub
```

And pull request:

```
git pull-request upstream 9.0
```

Nessesary to add some header for pull request. Save it. If everything is ok you will got link to your pull request.

## 7.7 Check remote bundings

Check current branch:

```
git branch -vv
```

Local branch must be bind to origin. If its no do next:

```
git push -u origin 9.0-pos_ms
```

## 7.8 Files relocation

- *git format-patch*
- *git filter-branch*

### 7.8.1 git format-patch

This section is based on OCA's [instruction](#).

Used variables:

- \$REPO\_PATH, \$REPO\_NAME - source repository
- \$MODULE - the name of the module you want to move
- \$BRANCH - the branch of the \$REPO with \$MODULE
- \$DEST\_REPO\_PATH, \$DEST\_REPO\_NAME - target repository

```
# Set variables
export REPO_PATH=/path/to/misc-addons REPO_NAME=misc-addons MODULE=some_module
$BRANCH=10.0 DEST_REPO_PATH=/path/to/mail-addons DEST_REPO_NAME=mail-addons

# Create patch
cd $REPO_PATH
git fetch upstream
git format-patch --stdout upstream/$BRANCH -- $MODULE > /tmp/relocation.patch

# Remove module from source repository
git checkout -b $BRANCH-$MODULE-relocation-remove upstream/$BRANCH
git rm -r $MODULE
git commit -m "[REM] $MODULE is relocated to $DEST_REPO_NAME"
git push origin
# then create PR on github

# Add commits to target repository
cd $DEST_REPO_PATH
git fetch upstream
git checkout -b $BRANCH-$MODULE-relocation-add upstream/$BRANCH
git am -3 < /tmp/relocation.patch
git push origin
# then create PR on github
```

### 7.8.2 git filter-branch

This section is based on <http://gbayer.com/development/moving-files-from-one-git-repository-to-another-preserving-history/>

**Goal:**

- Move directory 1 from Git repository A to Git repository B.

**Constraints:**

- Git repository A contains other directories that we don't want to move.

- We'd like to preserve the Git commit history for the directory we are moving.

### Let's start

- \$REPO: the repository hosting the module (e.g. misc-addons)
- \$DEST\_REPO: the repository you want to move the module to (e.g. access-addons)
- \$MODULE: the name of the module you want to move (e.g. group\_menu\_no\_access)
- \$BRANCH: the branch of the \$REPO with \$MODULE (source branch, e.g. 8.0)

**Warning:** If you have installed git from official ubuntu 14.04 deb repository then you should first update it. You can update git using this instruction [Update git](#)

```
$ cd ~
$ git clone https://github.com/it-projects-llc/$REPO -b $BRANCH
$ cd $REPO
$ git remote rm origin
$ git filter-branch --subdirectory-filter $MODULE -- --all
$ mkdir $MODULE
$ mv * $MODULE # never mind the "mv: cannot move..." warning message
$ git add .
$ git commit -m "[MOV] $MODULE: ready"
$ cd ~
$ cd $DEST_REPO
$ git remote add $MODULE-hosting-remote ~/ $REPO
$ git pull $MODULE-hosting-remote $BRANCH
```

After the last command you will have the module with all its commits in your destination repo. Now you can push it on github etc. You can remove ~/ \$REPO folder - no use of it now.

**Warning:** Cloning - this is required step. It is temporary directory. It will removed all modules except the one that you want to move.

The following script may come in handy if you need to move several modules. But be sure that you understand all its commands before using.

```
#!/bin/bash

source_repo=$PWD
echo $source_repo

if [ -n "$1" ]
then
    module=$1
    echo $module
else
    echo "Must be module name"
    exit $E_WRONGARGS
fi

if [ -n "$2" ]
then
```

(continues on next page)

(continued from previous page)

```

dest_repo=$2
echo $dest_repo
else
    echo "Must be dest_repo"
    exit $E_WRONGARGS
fi

if [ -n "$3" ]
then
    branch=$3
    echo $branch
else
    echo "Must be branch specified"
    exit $E_WRONGARGS
fi

cp -r $source_repo ../$module
cd ../$module
git remote rm origin
git filter-branch --subdirectory-filter $module -- --all
mkdir $module
mv * $module
git add .
git commit -m "[MOV] module -- $module"
cd $dest_repo
git remote add repo_moved_module $source_repo/../$module
git pull repo_moved_module $branch --no-edit
git remote rm repo_moved_module
rm -rf $source_repo/../$module

```

In order to use it you should make the movemode.sh file in your home directory and put all lines above there and make this file executable.

```
$ cd ~
$ chmod +x movemode.sh
```

To do the moving of group\_menu\_no\_access from addons-yelizariev to access-addons with the movemode.sh take the following steps.

```
$ cd ~
$ git clone https://github.com/yelizariev/addons-yelizariev.git
$ cd addons-yelizariev
```

This part is the same as moving without the script. But then I type only one command instead of many in case of fully manual approach.

```
addons-yelizariev$ ~/movemode.sh group_menu_no_access ~/access-addons 8.0
```

## 7.9 Git stash

- book: <https://git-scm.com/book/no-nb/v1/Git-Tools-Stashing>
- man: <https://git-scm.com/docs/git-stash>

## 7.10 Update Git

Ubuntu 14.04 official deb repository has 1.9 version of Git. It is too old and have to be updated.

<http://askubuntu.com/questions/579589/upgrade-git-version-on-ubuntu-14-04>

```
sudo apt-get remove git
sudo add-apt-repository ppa:git-core/ppa
sudo apt-get update
sudo apt-get install git
git --version
```

## 7.11 Squash commits into one

### 7.11.1 Backup

Before making a squash consider to “backup” your commits.

Local backup:

```
git tag 9.0-new-module-backup
```

Remote backup

```
git push origin 9.0-new-module:9.0-new-module-backup
```

To restore original state you can use following command:

```
# be sure that you on the branch you are going to change
git status

# restore from tag
git rebase 9.0-new-module-backup -X theirs

# restore from remote branchtag
git rebase origin/9.0-new-module-backup -X theirs
```

### 7.11.2 git commit --amend

Instead of creating new commit, adds updates to the latest commit.

### 7.11.3 git rebase -i

Interactive squashing

```
git rebase -i <your-first-commit>^
# e.g.
git rebase -i 7801c8b^
```

Then edit opened file and keep pick for the first commit and replace pick with squash for the rest ones. E.g.

Origin:

```
TODO
```

Edited:

```
TODO
```

**Warning:** If you remove a line here THAT COMMIT WILL BE LOST.

#### 7.11.4 Push

```
git push -f origin 9.0-new-module
```

### 7.12 Create branch from another's Pull Request

```
git fetch upstream pull/354/head:pr354
git checkout -b 10.0-branch-name pr354
```

More information: <https://help.github.com/articles/checking-out-pull-requests-locally/>

#### 7.12.1 Push updates to another's Pull Request

If you have access to edit PR files via github UI, you can push such updates from console

```
GITHUB_USERNAME=yelizariev # set username where PR is made from
REPO=pos-addons # set repo name
BRANCH=10.0-fix-something # set source branch name

git remote add ${GITHUB_USERNAME} git@github.com:${GITHUB_USERNAME}/${REPO}.git
git fetch ${GITHUB_USERNAME} ${BRANCH}
git checkout ${GITHUB_USERNAME}/${BRANCH}
# make updates
# ...
# make commit
git commit ...

# push update to another's branch
git push ${GITHUB_USERNAME} HEAD:${BRANCH}
```



## Continuous Integration

### 8.1 Runbot

- [runbot.odoo.com](http://runbot.odoo.com)
  - [How to use runbot.odoo.com?](#)
- [runbot.it-projects.info](http://runbot.it-projects.info)
- [How to deploy runbot?](#)

#### 8.1.1 runbot.odoo.com

<http://runbot.odoo.com/> – official runbot. While its main purpose is checking pull requests to official repository, it is useful on daily development routine.

- It allows to play with any odoo version. Each build has all modules installed with demo data.
- It allows to quickly try enterprise odoo versions

##### How to use runbot.odoo.com?

- open <http://runbot.odoo.com/runbot/>
- switch to repository you need. Odoo community (odoo/odoo) is default.
- find a row with odoo version you need (10.0, 9.0, 8.0, 7.0)
- click on *fast forward* icon to open latest build. Alternatively, click on any blue button on a row, that corresponds to odoo version you need.
- on login page enter credentials:

- Admin
  - \* login: admin
  - \* password: admin
- Demo
  - \* login: demo
  - \* password: demo

The screenshot shows the runbot interface for Odoo development. At the top, there are tabs for different Odoo versions: odoo/odoo, odoo-dev/odoo, odoo/design-themes, odoo/enterprise, and odoo-dev/enterprise. Below the tabs, a search bar and a 'Pending: 0' indicator are visible.

The main area displays build results for four Odoo branches:

- 10.0 8h**: [FIX] website\_sale: typo in check... by Jeremy Kersten (17047-10-0-651738 on runbot10.odoo.com)
- 7.0 8d**: [FIX] Fix error in error message... by Thibault Fauvelois (175769-7-0-34c1a on runbot5.odoo.com)
- 8.0 16h**: [IMP] Allow configuration of ... by Leonardo Rochael Almeida (177011-8-0-5ee434 on runbot9.odoo.com)
- 9.0 1m**: [FIX] point\_of\_sale: create two ... by Wolfgang Taferner (177007-9-0-19cc5 on runbot8.odoo.com)

Each card contains a summary of the fix, the author's name, the commit ID, and the time taken for the build. Below the cards, there are sections for packaging and web-related fixes. A red arrow points to the 'Branch or pull' dropdown at the top left, another to the 'Branch or pull' button on the first card, and a third to the 'Branch or pull' dropdown at the bottom left.

### 8.1.2 runbot.it-projects.info

<http://runbot.it-projects.info/> – customized runbot for IT-Projects' repositories.

Stages of making a build:

- Checkout sources from github
- **-base** database: install updated modules for pull request builds and base modules for branch builds. For some repositories explicit modules (i.e. ones, that are specified in runbot settings) are installed too
- **-all** database: install all modules of the repo
- run the build with two prepared databases

Main features:

- Blue button - enter to **-all** database
- Green button - enter to **-base** database
- Key logs (shown on build page) – key logs, warnings and errors
- Detailed logs (txt files)
  - *Full base logs* – full logs of installation process in **-base** database
  - *Full all logs* – full logs of installation process in **-all** database

- *Full run logs* – full logs for both databases after running, i.e. when Blue and Green button are available. Logs includes cron work, url requests etc

The screenshot shows the Odoo Travis CI interface. At the top, there's a header with the Odoo logo, the repository URL "github.com/it-projects-llc/misc-addons", and a build ID "00782-8-0-a69575". Below this, the commit hash "a69575" is shown. To the right, there are several icons: a blue arrow, a green arrow, a file icon, a circular arrow icon, and a gear icon. A red box highlights the "Logs" section, which contains three options: "Full base logs", "Full all logs", and "Full run logs". Red arrows point from the "branch name or pull number" and "commit hash" labels to their respective fields in the UI. Another red arrow points from the "detailed logs" label to the "Logs" section. A red box also surrounds the "Key logs" table.

Date	Level	Type	Message	Branch or pull
2016-10-25 08:54:45	INFO	runbot	init Init build environment	Commit
2016-10-25 08:54:45	INFO	runbot	checkout closest branch for odoo/odo	Compare
2016-10-25 08:54:45	INFO	runbot	checkout closest branch for it-projects	Runtime: 3ms
2016-10-25 08:54:45	INFO	runbot	checkout closest branch for it-projects	Port: 2000
2016-10-25 08:54:45	INFO	runbot	checkout closest branch for it-projects	Age: 20h
2016-10-25 08:54:45	INFO	runbot	checkout closest branch for it-projects-llc/mail-addons is refs/heads/8.0	/heads/8.0
2016-10-25 08:54:45	INFO	runbot	checkout closest branch for it-projects-llc/server-tools is refs/heads/8.0	
2016-10-25 08:54:45	INFO	runbot	checkout closest branch for it-projects-llc/website is refs/heads/8.0	
2016-10-25 08:54:45	INFO	runbot	checkout closest branch for it-projects-llc/project is refs/heads/8.0	
2016-10-25 08:54:45	INFO	runbot	checkout closest branch for it-projects-llc/hr is refs/heads/8.0	

**Key logs**

### 8.1.3 How to deploy runbot?

There is docker that allows you deploy your own runbot for your repositories. Check it out for further information

- <https://github.com/it-projects-llc/odoo-runbot-docker>

## 8.2 Odoo Travis Tests

TODO

### 8.3 Coverage



## 9.1 Models

Section helps in understanding built-in models

### 9.1.1 ir.config\_parameter

**Add record by module**

**XML: <record>**

Code:

```
<data noupdate="1">
    <record id="myid" model="ir.config_parameter">
        <field name="key">mymodule.mykey</field>
        <field name="value">True</value>
        <field name="group_ids" eval="[(4, ref('base.group_system'))]"/>
    </record>
```

Prons:

- record is deleted on uninstalling

Cons:

- it raises error, if record with that key is already created manually

**XML: <function>**

Code:

```
<function model="ir.config_parameter" name="set_param" eval="('auth_signup.allow_uninvited', True, ['base.group_system'])" />
```

Prons:

- it doesn't raise error, if record with that key is already created manually

Cons:

- record is not deleted on uninstalling
- value is overwritten after each module updating

## YML

Code:

```
- !python {model: ir.config_parameter}: |
    SUPERUSER_ID = 1
    if not self.get_param(cr, SUPERUSER_ID, "ir_attachment.location"):
        self.set_param(cr, SUPERUSER_ID, "ir_attachment.location", "postgresql:lobject")
```

Prons:

- value is not overwritten if it already exists

Cons:

- record is not deleted on uninstalling

### 9.1.2 res.users

TODO

### 9.1.3 res.groups

TODO

### 9.1.4 ir.model.access

Defines access to a whole model.

Each access control has a model to which it grants permissions, the permissions it grants and optionally a group.

Access controls are additive, for a given model a user has access all permissions granted to any of its groups: if the user belongs to one group which allows writing and another which allows deleting, they can both write and delete.

If no group is specified, the access control applies to all users, otherwise it only applies to the members of the given group.

Available permissions are creation (`perm_create`), searching and reading (`perm_read`), updating existing records (`perm_write`) and deleting existing records (`perm_unlink`)

When there is no access records for a given model and a permission (e.g. `read`), then only Superuser has the permission.

See also:

- [Superuser rights](#)
- [ir.rule](#)

## Fields

```
name = fields.Char(required=True, index=True)
active = fields.Boolean(default=True, help='If you uncheck the active field, it will disable the ACL without deleting it (if you delete a native ACL, it will be re-created when you reload the module).')
model_id = fields.Many2one('ir.model', string='Object', required=True, domain=[('transient', '=', False)], index=True, ondelete='cascade')
group_id = fields.Many2one('res.groups', string='Group', ondelete='cascade', index=True)
perm_read = fields.Boolean(string='Read Access')
perm_write = fields.Boolean(string='Write Access')
perm_create = fields.Boolean(string='Create Access')
perm_unlink = fields.Boolean(string='Delete Access')
```

### 9.1.5 ir.rule

Record rules are conditions that records must satisfy for an operation (create, read, write or delete) to be allowed. Example of a condition: *User can update Task that assigned to him.*

Group field defines for which group rule is applied. If Group is not specified, then rule is *global* and applied for all users.

Domain field defines conditions for records.

Boolean fields (read, write, create, delete) of ir.rule mean *Apply this rule for this kind of operation.* They do **not** mean *restrict access for this kind of operation.*

## Checking access algorithm

To check either user has access for example to *read* a record, system do as following:

- Check access according to [ir.model.access](#) records. If it doesn't pass, then user **doesn't get** access
- Find and check global rules for the **model** and for *read* operation
  - if the record **doesn't satisfy** (doesn't fit to domain) for at least one of the global rules, then user **doesn't get** access
- Find and check non-global rules for the **model** and for *read* operation.
  - if there are no such groups, then user **get** access
  - if the record **satisfy** (fit to domain) for **at least one** of the non-global rules, then user **get** access
  - if the record **doesn't satisfy** for **all** non-global rules, then user **doesn't get** access

See also:

- [Superuser rights](#)

## Fields

```
name = fields.Char(index=True)
active = fields.Boolean(default=True, help="If you uncheck the active field, it will
    ↪ disable the record rule without deleting it (if you delete a native record rule, it
    ↪ may be re-created when you reload the module).")
model_id = fields.Many2one('ir.model', string='Object', index=True, required=True,
    ↪ ondelete="cascade")
groups = fields.Many2many('res.groups', 'rule_group_rel', 'rule_group_id', 'group_id')
domain_force = fields.Text(string='Domain')
domain = fields.Binary(compute='_force_domain', string='Domain')
perm_read = fields.Boolean(string='Apply for Read', default=True)
perm_write = fields.Boolean(string='Apply for Write', default=True)
perm_create = fields.Boolean(string='Apply for Create', default=True)
perm_unlink = fields.Boolean(string='Apply for Delete', default=True)
```

## 9.1.6 product.template

The stores have products that differ from some other only a one or few properties. Such goods it makes no sense to separate as individual products. They are join in a group of similar goods, which are called **template**.

**shop:** product pages use product.template (when order is created, then *product.product* is used).

## 9.1.7 product.product

The product, unlike the *template*, it is a separate product that can be calculated, set the price, to assign a discount.

*product.product* is used:

- sale.order
- stock
- pos

## 9.1.8 ir.actions.todo

The model is used for executing actions (records in the “ir.actions.act\_window” model). The model allows to set conditions and sequence of appearance of wizards. Also you can specify a regular interface window but only as last action. Code:

```
<record id="sce.initial_setup" model="ir.actions.todo">
    <field name="action_id" ref="action_initial_setup"/>
    <field name="state">open</field>
    <field name="sequence">1</field>
    <field name="type">automatic</field>
</record>
```

The startup type can be one of the following:

- manual: Launched manually.
- automatic: Runs whenever the system is reconfigured. The launch takes place either after install/upgrade any module or after calling the “execute” method in the “res.config” model.
- once: After having been launched manually, it sets automatically to Done.

## 9.1.9 bus.bus

### Bus

Bus is a module for instant notifications via longpolling. Add it to dependencies list:

```
'depends': ['bus']
```

---

**Note:** Mail module in odoo 9.0 is already depended on module bus.

---

**Warning:** Don't mistake longpolling bus with `core.bus` which is client-side only and part of `web` module.

### What is longpolling

- *About longpolling*
- *How to enable Longpolling in odoo*

### How to implement longpolling

- *Scheme of work*
  - *Channel identifier*
  - *Listened channels*
  - *Binding notification event*
  - *Start polling*
  - *Sending notification*
  - *Handling notifications*

### Scheme of work

- Specify channels that current client is listening
- Bind notification event to your handler
- Start polling
- Send notification to some channel via python code

### Channel identifier

Channel identifier - is a way to distinguish one channel from another. In the main, channel contains dbname, some string and some id.

Added via js identifiers can be string only.

```
var channel = JSON.stringify([dbname, 'model.name', uid]);
```

Added via python identifiers can be a string or any data structure.

```
# tuple
channel = (request.db, 'model.name', request.uid)
# or a string
channel = '[ "%s", "%s", "%s"]' % (request.db, 'model.name', request.uid)
```

**Warning:** JSON.stringify in js and json.dumps in python could give a different result.

### Listened channels

You can add channels in two ways: either on the server side via `_poll` function in bus controller or in js file using the method `bus.add_channel()`.

With controllers:

```
# In odoo 8.0:
import openerp.addons.bus.bus.Controller as BusController

# In odoo 9.0:
import openerp.addons.bus.controllers.main.BusController

class Controller(BusController):
    def _poll(self, dbname, channels, last, options):
        if request.session.uid:
            registry, cr, uid, context = request.registry, request.cr, request.
            session.uid, request.context
            new_channel = (request.db, 'module.name', request.uid)
            channels.append(new_channel)
        return super(Controller, self). _poll(dbname, channels, last, options)
```

In the js file:

```
// 8.0
var bus = openerp.bus.bus;
// 9.0+
var bus = require('bus.bus').bus;

var channel = JSON.stringify([dbname, 'model.name', uid]);
bus.add_channel(new_channel);
```

### Binding notification event

In js file:

```
bus.on("notification", this, this.on_notification);
```

## Start polling

In js file:

```
bus.start_polling();
```

---

**Note:** You don't need to call `bus.start_polling()`; if it was already started by other module.

---

When polling starts, request `/longpolling/poll` is sent, so you can find and check it via Network tool in your browser

## Sending notification

You can send notification only through a python. If you need to do it through the client send a signal to server in a usual way first (e.g. via controllers).

```
self.env['bus.bus'].sendmany([(channel1, message1), (channel2, message2), ...])
# or
self.env['bus.bus'].sendone(channel, message)
```

## Handling notifications

```
on_notification: function (notifications) {
    // Old versions passes single notification item here. Convert it to the latest_
    ↪format.
    if (typeof notification[0][0] === 'string') {
        notification = [notification]
    }
    for (var i = 0; i < notification.length; i++) {
        var channel = notification[i][0];
        var message = notification[i][1];

        // proceed a message as you need
        // ...
    }
},
```

## Examples

### pos\_multi\_session:

- add channel (python)
- bind event
- send notification

### chess:

- add channel (js)
- bind event
- send notification

**mail\_move\_message:**

- add channel (python)
- bind event
- send notification

## 9.1.10 ir.cron

### Creating automated actions in Odoo

Schedulers are automated actions that run automatically over a time period and can do a lot of things. They give the ability to execute actions database without needing manual interaction. Odoo makes running a background job easy: simply insert a record to `ir.cron` table and Odoo will execute it as defined.

#### 1. Creating the model and method of this model.

```
class model_name(models.Model):  
    _name = "model.name"  
    # fields  
    def method_name(self, cr, uid, context=None): # method of this model  
        # your code
```

#### 2. Creating the automated action

*If you want to build new modules in the guidelines from Odoo you should add the code for an automated action under `yourDefaultModule/data/` in a separate XML file.*

An important thing to note with automated actions is that they should always be defined within a `noupdate` field since this shouldn't be updated when you update your module.

```
<openerp>  
    <data noupdate="1">  
        <record id="unique_name" model="ir.cron">  
            <field name="name">Name </field>  
            <field name="active" eval="True" />  
            <field name="user_id" ref="base.user_root" />  
            <field name="interval_number">1</field>  
            <field name="interval_type">days</field>  
            <field name="numbercall">-1</field>  
            <field name="doall">1</field>  
            <!--<field name="nextcall" >2016-12-31 23:59:59</field>-->  
            <field name="model" eval="'model.name'" />  
            <field name="function" eval="'method_name'" />  
            <field name="args" eval="" />  
            <!--<field name="priority" eval="5" />-->  
        </record>  
    </data>  
</openerp>
```

The first thing you notice is the `data noupdate="1"`, this is telling Odoo that all code within this tag shouldn't be updated when you update your module.

```
<record id="unique_name" model="ir.cron">
```

The `id` is an unique identifier for Odoo to know what record is linked to which `id`. The `model` called ("ir.cron") is the model specifically made by Odoo for all automated actions. This model contains all automated actions and should always be specified.

```
<field name="name">Name </field>
```

The next line is the name.

```
<field name="active" eval="True" />
```

Boolean value indicating whether the cron job is active or not.

```
<field name="user_id" ref="base.user_root"/>
```

This user id is referring to a specific user, in most cases this will be base.user\_root.

```
<field name="interval_number">1</field>
```

Number of times the scheduler is to be called based on the “interval\_type”

```
<field name="interval_type">days</field>
```

Interval Unit.

It should be one value for the list: minutes, hours, days, weeks, months.

```
<field name="numbercall">-1</field>
```

An integer value specifying how many times the job is executed. A negative value means no limit.

```
<field name="doall">1</field>
```

A boolean value indicating whether missed occurrences should be executed when the server restarts.

```
<field name="nextcall" >2016-12-31 23:59:59</field> <!-- notice the date/time format -->
```

Next planned execution date for this job.

```
<field name="model" eval="'model.name'" />
```

The field model specifies on which model the automated action should be called.

```
<field name="function" eval="'method_name'" />
```

Name of the method to be called when this job is processed.

```
<field name="args" eval="" />
```

The arguments to be passed to the method.

```
<field name="priority" eval="5" />
```

The priority of the job, as an integer: 0 means higher priority, 10 means lower priority.

## Defaults.

Name	Definition
nextcall	lambda *a: time.strftime(DEFAULT_SERVER_DATETIME_FORMAT)
priority	5
user_id	lambda obj,cr,uid,context: uid
interval_number	1
interval_type	months
numbercall	1
active	1
doall	1

## 9.1.11 mail.message

### Message Subtypes in Odoo

Most of the time in Odoo multiple users work upon one particular record or document like sale order,Invoice ,Tasks etc. In such scenarios,it becomes extremely important to track changes done by every individual against that document. It helps management to find any possible reason in case of any issue occurs. Odoo provides this feature to great extent with the help of OpenChatter Integration.

Consider a scenario where multiple users are working in a single project.Various parameters for that project are already configured like deadline,Initially Planned Hours etc. Now one of the user changes the value of Planned Hours. So now it is important to know which user has changed it and what was the previous value. We can track it by creating message subtypes in Odoo as following.

It needs to be defined in XML which will have following syntax.

```
<record id="mt_task_planned_hours" model="mail.message_subtype">
    <field name="name">Task planned hours changed</field>
    <field name="res_model">project.task</field>
    <field name="default" eval="True"/>
    <field name="description">Task planned hours changed</field>
</record>
```

Users can also have a mail.message.subtype that depends on an other to act through a relation field. For the planned hours, we can have following syntax for it.

```
<record id="mt_task_planned_hours_change" model="mail.message_subtype">
    <field name="name">Task planned hours changed</field>
    <field name="sequence">10</field>
    <field name="res_model">project.project</field>
    <field name="parent_id" eval="ref('mt_task_planned_hours')"/>
    <field name="relation_field">project_id</field>
</record>
```

Odoo provide feature to track various events related with one particular document with the help of \_track attribute. If we inherit mail.thread object then with the help of \_track attribute, user can sent notification also to keep aware others about changes happening against this particular document. The syntax can be as follow.

```
_track = {
    'planned_hours': {
        'project.mt_task_planned_hours': lambda self, cr, uid, obj, ctx=None: obj.planned_
        ↪hours,
    },
}
```

In order to track changes related with any field,Odoo provides an attribute named as track\_visibility.It has to be defined at field level which has below syntax.

```
planned_hours = fields.Float(string = 'Initially Planned Hours', track_visibility='onchange', help='Estimated time to do the task, it is project manager when the task is in draft state.')
```

Hence, it is easy to track the changes done so far against any particular document by different users.

## 9.2 How to use Odoo

### 9.2.1 How to create database

#### From UI

To create new database open /web/database/manager

**8.0-**

#### Database with dots

Early version of odoo doesn't allow to create databases with dots. You can remove this restriction in two ways:

1. Updates sources:

```
cd path/to/odoo
sed -i 's/matches="[^"]*"/g' addons/web/static/src/xml/base.xml
```

2. update html code via *Inspect Element* tool

You must remove the matches field value.



#### From terminal

**9.0+**

To create new database simple add -d parameter when you run odoo, e.g.:

```
./openerp-server -d database1
```

– will create new database with name database1

## 9.2.2 How to enable Technical Features

### 8.0

- open Settings / Users / Users
- select your user
- click [Edit]
- switch Technical Features on
- click [Save]
- refresh web page (click F5)

### 9.0+

Since Odoo 9.0 to enable Technical Features you only need to *activate developer mode*.

## 9.2.3 How to install/update module

- *From zip archive*

- 11.0+
  - \* *install*
  - \* *update*
- 10.0+
  - \* *install*
  - \* *update*
- 9.0
  - \* *install*
  - \* *update*
- 8.0
  - \* *install*
  - \* *update*

**Warning:** Import Module tool (import from a zip file) doesn't work for modules with python files. It means that it doesn't work in most cases

Apps    Apps    App Store    Updates    Update Apps List    Apply Scheduled Upgrades    Import Module

Administrator (import\_test)

Apps    Search...

``Import Module`` tool doesn't work for modules with python files 1-52 / 52

CRM Leads, Opportunities, Activities <code>crm</code>	Project Projects, Tasks <code>project</code>	Inventory Management Inventory, Logistics, Warehousing <code>stock</code>
Manufacturing Manufacturing Orders, Bill of Materials, Routings <code>mrp</code>	Sales Management Quotations, Sales Orders, Invoicing <code>sale_management</code>	Point of Sale Touchscreen Interface for Shops <code>point_of_sale</code>

Discuss    Apps    Settings    Administrator (import\_t...)

odoo

Apps  
App Store  
Updates  
Update Apps List  
Apply Scheduled Upgrades  
Import Module

Apps    Search...

``Import Module`` tool doesn't work for modules with python files.

CRM Leads, Opportunities, Activities <code>crm</code>	Project Projects, Tasks <code>project</code>
Inventory Management Inventory, Logistics, Warehousing <code>stock</code>	Manufacturing Manufacturing Orders, Bill of Materials, Routings <code>mrp</code>

## From zip archive

- unzip module to your addons folder
- restart odoo server

## 11.0+

### install

- *activate developer mode*
- navigate to Apps menu
- click Update Apps List
- search and open a module you need
- click [Install]

### update

- navigate to Apps menu
- search and open a module you need

- click [Upgrade]

## 10.0+

### install

- *activate developer mode*
- navigate to Apps menu
- click Update Apps List
- search and open a module you need
- click [Install]

### update

- navigate to Apps menu
- search and open a module you need
- click [Upgrade]

## 9.0

### install

- *activate developer mode*
- navigate to Apps menu
- click Update Apps List
- search and open a module you need
- click [Install]

### update

- navigate to Apps menu
- search and open a module you need
- click [Upgrade]

## 8.0

### install

- navigate to [ [ Settings ] ] >> Local Modules
- search and open a module you need
- click [Install]

## update

- navigate to [ [ Settings ] ] >> Local Modules
- search and open a module you need
- click [Upgrade]

### 9.2.4 How to activate developer mode

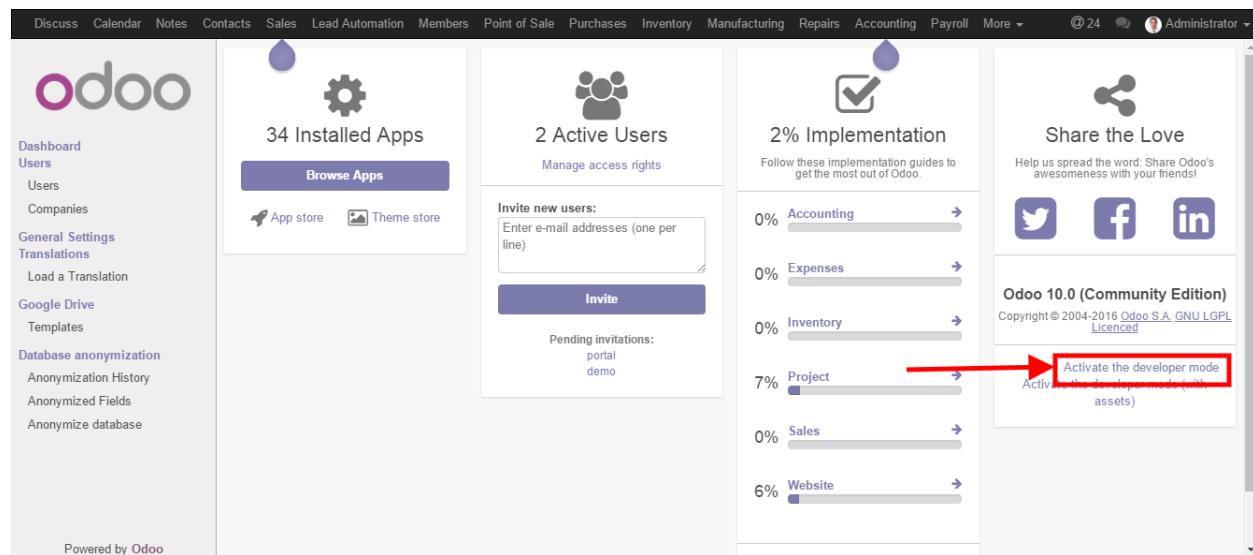
Add debug parameter to your url, for example:

```
localhost:8069/web?debug=1
```

or use UI as described below

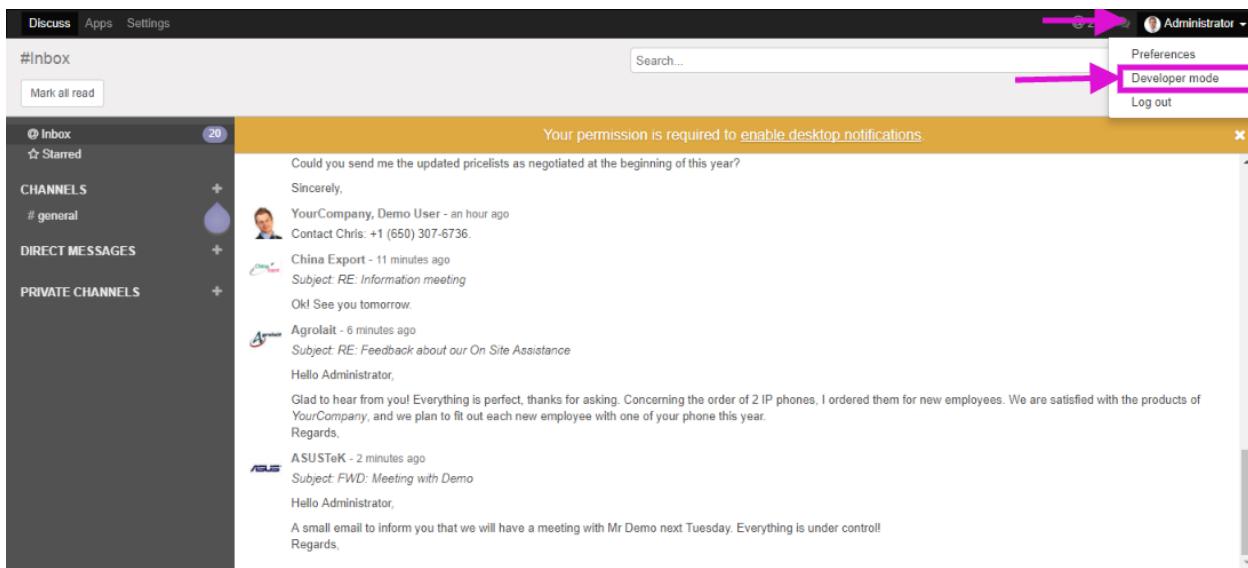
#### 10.0+

- go to Settings
- click Activate the developer mode



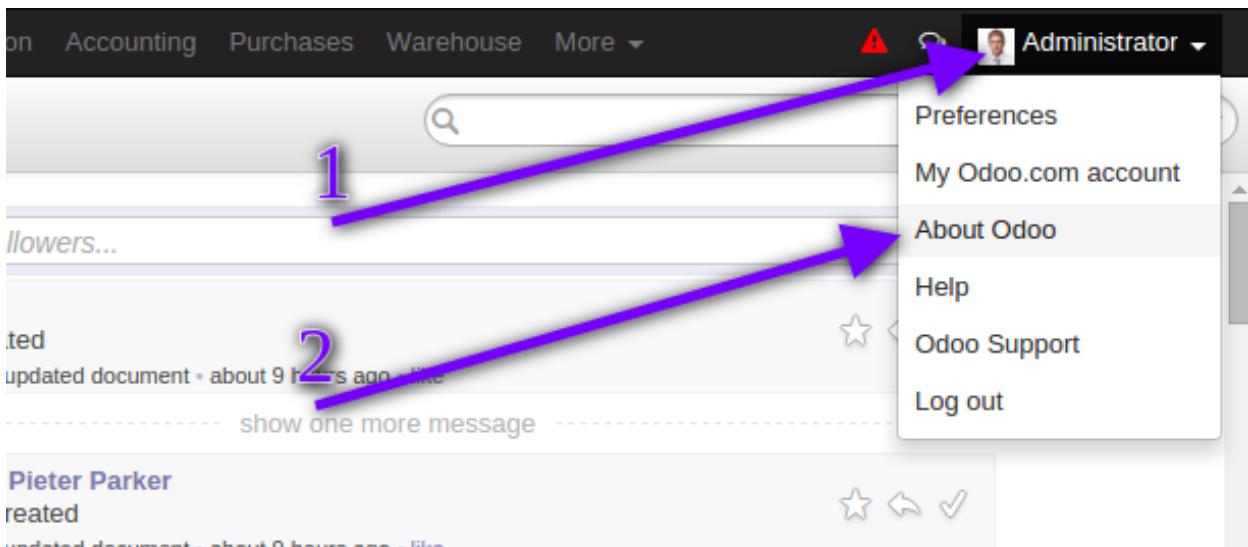
#### 10.0+ with the web\_debranding

- go to the User menu on the upper right corner
- click Developer mode

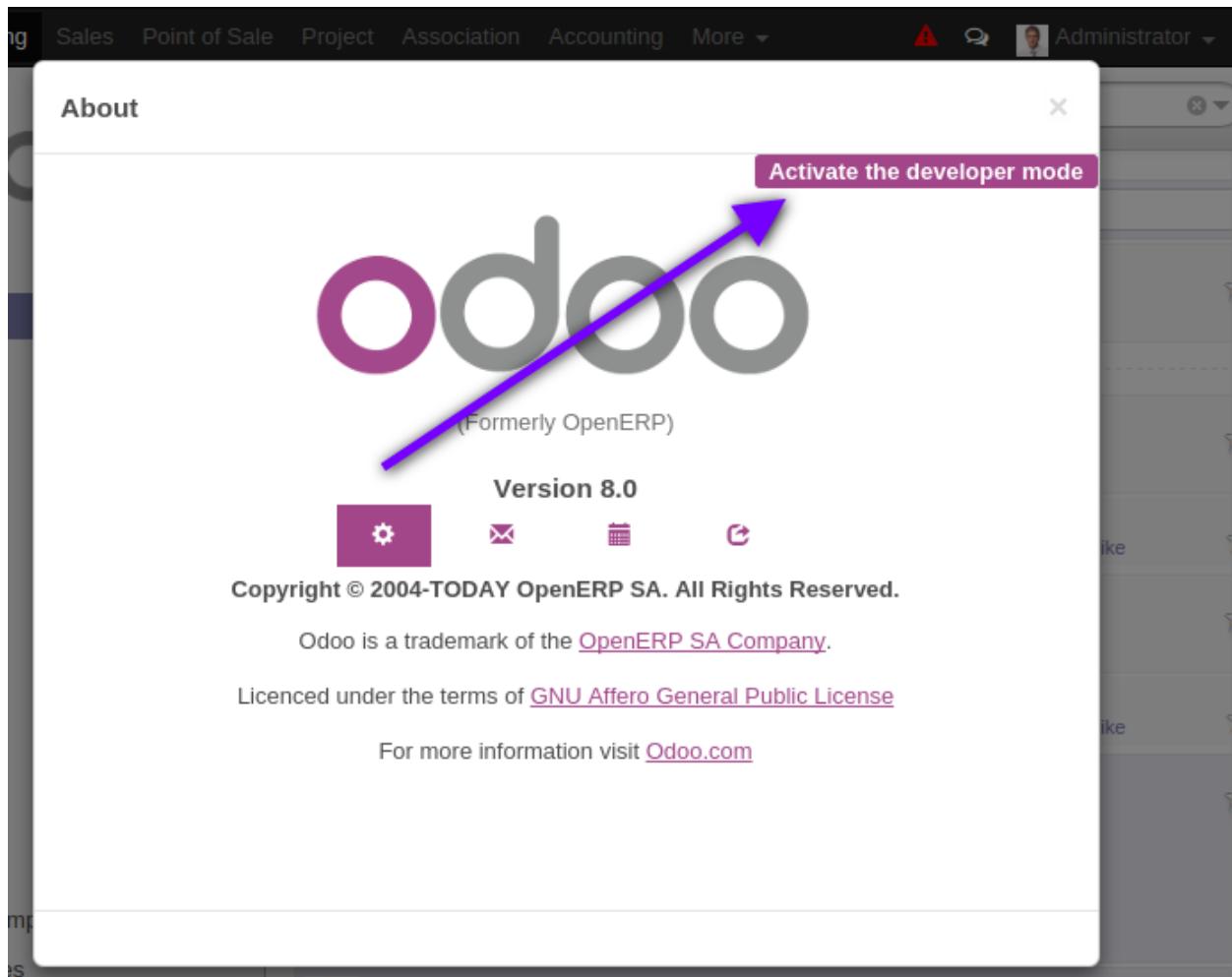


### 9.0, 8.0

- click button at top right-hand corner <User Name> -> About Odoo



- click Activate the developer mode



- In odoo 8.0 you may need to *Enable technical features* too

### 9.2.5 How to activate debug assets mode

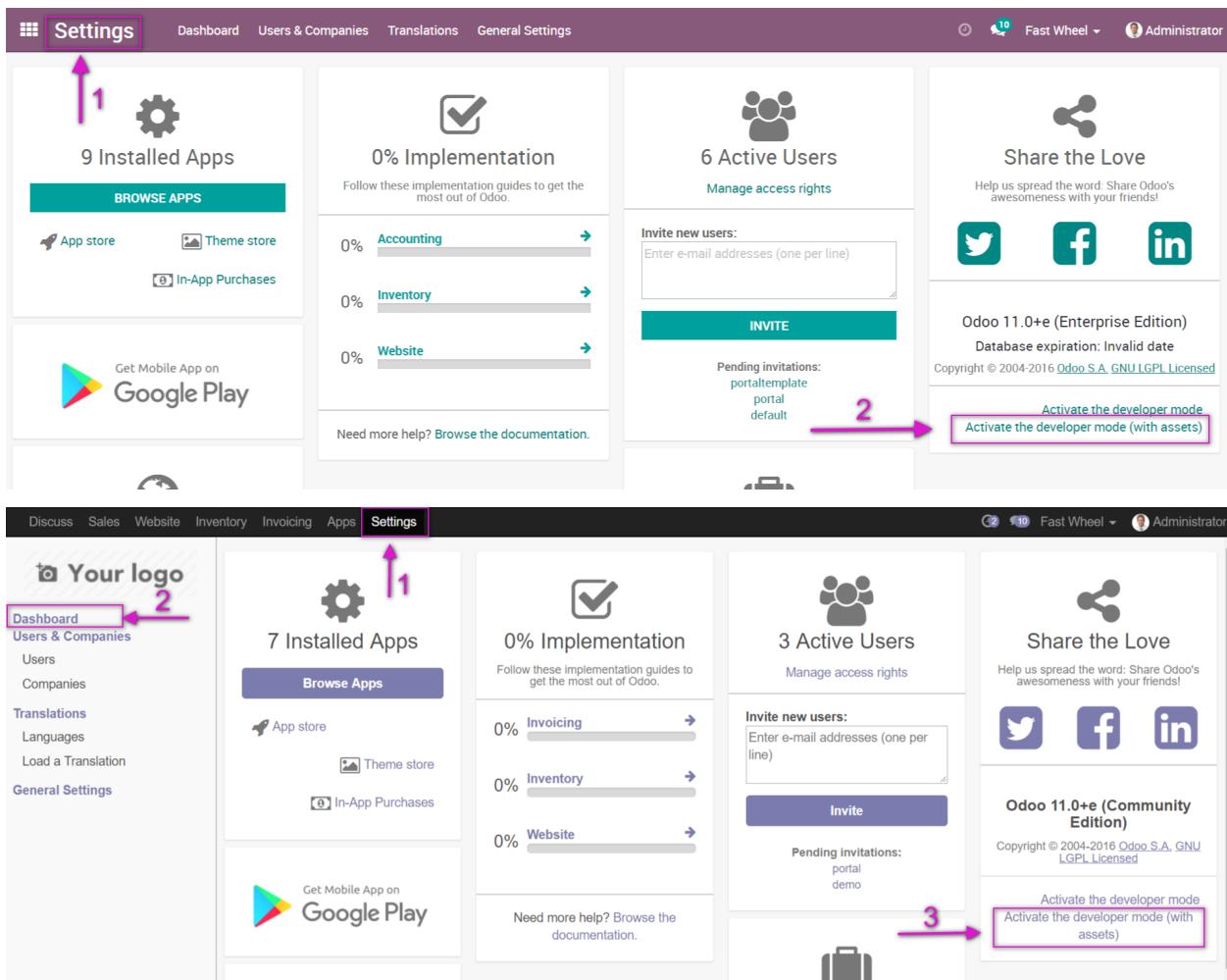
Add `debug=assets` parameter to your url, e.g.:

```
localhost:8069/web?debug=assets#
```

or use UI as described below

#### 10.0+

- go to Settings
- click Activate the developer mode (with assets)



### 9.2.6 How to log in as a superuser

**12.0+**

- open page with login form, for example:

```
localhost:8069/web/login
```

- add `debug` parameter to your url, for example:

```
localhost:8069/web/login?debug=1
```

- enter the username and password of a user which is in Administration: Settings (base\_group\_system) security group, for example:

```
Username: admin  
Password: admin
```

- click to Log in as superuser link below the Log in button

Email

admin

Password

\*\*\*\*\*

Log in

Log in as superuser





# CHAPTER 10

---

## Odoo administration

---

Official docs:

- <https://www.odoo.com/documentation/8.0/setup/install.html>
- <https://www.odoo.com/documentation/8.0/setup/deploy.html>

---

**Note:** This article is obsolete and will be removed soon

---

### 10.1 Odoo installation

#### Contents

- *Odoo installation*
  - *Docker installation*
    - \* *Install docker*
    - \* *Clone repositories*
    - \* *Create dockers*
    - \* *Control dockers*
  - *Straightforward installation*
  - *Nginx configuration*

---

**Note:** This article covers development installation only. For production installation follow <https://github.com/it-projects-llc/install-odoo>

---

### 10.1.1 Docker installation

#### Install docker

Follow <https://docs.docker.com/engine/installation/>

#### Clone repositories

```
cd /some/work/path

## Settings
ODOO_BRANCHES=(11.0 10.0 9.0 8.0) # update if needed
GITHUB_USER=yelizariev # change it to your user

## Common functions
function init_repo {
    MAIN=$1
    NAME=$2
    if [ ! -d $NAME ]; then
        # clone
        git clone https://github.com/${MAIN}/${NAME}.git $NAME
        # rename
        git -C $NAME remote rename origin upstream
    fi

    # NAME origin over ssh
    git -C $NAME remote add origin git@github.com:${GITHUB_USER}/${NAME}.git

    for b in "${ODOO_BRANCHES[@]}"
    do
        DEST=odoo-$b/$NAME

        if [ ! -d $DEST ]; then
            # copy
            cp -r $NAME $DEST
            # checkout to branch
            git -C $DEST checkout upstream/$b
        fi
    done

    # clean up
    rm -rf $NAME
}

## Create folder odoo-$b for each branch
for b in "${ODOO_BRANCHES[@]}"
do
    mkdir -p odoo-$b
done

## Clone odoo
init_repo odoo odoo

## Clone IT_PROJECTS_LLC_REPOS
IT_PROJECTS_LLC_REPOS=(
```

(continues on next page)

(continued from previous page)

```

"pos-addons"
"access-addons"
"website-addons"
"misc-addons"
"mail-addons"
"odoo-saas-tools"
"odoo-telegram"
)

for r in "${IT_PROJECTS_LLC_REPOS[@]}"
do
    init_repo it-projects-llc $r
done

## Clone addons-dev
init_repo it-projects-llc addons-dev
for b in "${ODOO_BRANCHES[@]}"
do
    git -C odoo-$b/addons-dev/ remote add misc-addons https://github.com/it-
    ↪projects-llc/misc-addons.git
    git -C odoo-$b/addons-dev/ remote add pos-addons https://github.com/it-
    ↪projects-llc/pos-addons.git
    git -C odoo-$b/addons-dev/ remote add mail-addons https://github.com/it-
    ↪projects-llc/mail-addons.git
    git -C odoo-$b/addons-dev/ remote add access-addons https://github.com/it-
    ↪projects-llc/access-addons.git
    git -C odoo-$b/addons-dev/ remote add website-addons https://github.com/it-
    ↪projects-llc/website-addons.git
    git -C odoo-$b/addons-dev/ remote add l10n-addons https://github.com/it-
    ↪projects-llc/l10n-addons.git
done

```

## Create dockers

```

# Create postgres docker container.
# You create one per each odoo version or one per each project / module
DB_CONTAINER=db-odoo-10
docker run -d -e POSTGRES_USER=odoo -e POSTGRES_PASSWORD=odoo --name $DB_CONTAINER \
↪postgres:9.5

ODOO_CONTAINER=some-container-name-for-odoo-10
ODOO_BRANCH=10.0

# Create docker without adding folders from host machine.
# Usually for demonstration and testing, not for development.
docker run \
-p 8069:8069 \
-p 8072:8072 \
-e ODOO_MASTER_PASS=admin \
--name $ODOO_CONTAINER \
--link $DB_CONTAINER:db \
-t itprojectsllc/install-odoo:$ODOO_BRANCH

# Attach folder from host to make updates there (example for misc-addons).
# It also runs odoo with "-d" and "--db-filter" parameters to work only with one \
↪database named "misc".

```

(continues on next page)

(continued from previous page)

```

# It prevents running cron task on all available databases
# In this example you need to add misc.local to /etc/hosts and open odoo via http://
↪misc.local
docker run \
-p 8069:8069 \
-p 8072:8072 \
-e OODOO_MASTER_PASS=admin \
-v /some/path/at/host-machine/with/clone-of-misc-addons-or-addons-dev/:/mnt/addons/it-
↪projects-llc/misc-addons/ \
--name $ODOO_CONTAINER \
--link $DB_CONTAINER:db \
-t itprojectsllc/install-odoo:$ODOO_BRANCH -- -d misc --db-filter ^%d$

# Update all repos
docker exec -t $ODOO_CONTAINER /bin/bash -c "export GIT_PULL=yes; bash /install-odoo-
↪saas.sh"

# Update odoo only
docker exec -t $ODOO_CONTAINER git -C /mnt/odoo-source/ pull

# Update misc-addons only
docker exec -t $ODOO_CONTAINER git -C /mnt/addons/it-projects-llc/misc-addons pull

```

## Control dockers

```

# open docker terminal as odoo
docker exec -i -t $ODOO_CONTAINER /bin/bash

# open docker terminal as root
docker exec -i -u root -t $ODOO_CONTAINER /bin/bash

# watch logs
docker attach $ODOO_CONTAINER

# stop container
docker stop $ODOO_CONTAINER

# start container
docker start $ODOO_CONTAINER

# remove container (if you don't need one anymore or want to recreate it)
docker rm $ODOO_CONTAINER

```

### 10.1.2 Straightforward installation

**Warning:** This way is not recommended and script may be obsolete

```

sudo apt-get update
sudo apt-get install git python-pip htop moreutils tree nginx gimp wmctrl postgresql-
↪server-dev-all

```

(continues on next page)

(continued from previous page)

```

sudo apt-get upgrade

#####
# Github
# configure ssh keys: https://help.github.com/articles/generating-ssh-keys/

#####
# Odoo
# download odoo from git:
cd /some/dir/
git clone https://github.com/odoo/odoo.git

# install dependencies:
wget http://nightly.odoo.com/9.0/nightly/deb/odoo_9.0.latest_all.deb
sudo dpkg -i odoo_9.0.latest_all.deb # shows errors -- just ignore them and execute
# next command:
sudo apt-get -f install
sudo apt-get remove odoo

# install wkhtmltox
cd /usr/local/src
lsb_release -a
uname -i
# check version of your OS and download appropriate package
# http://wkhtmltopdf.org/downloads.html
# e.g.
apt-get install xfonts-base xfonts-75dpi
apt-get -f install
wget http://download.gna.org/wkhtmltopdf/0.12/0.12.2.1/wkhtmltox-0.12.2.1_linux-
# trusty-amd64.deb
dpkg -i wkhtmltox*.deb

# requirements.txt
cd /path/to/odoo
sudo pip install -r requirements.txt
sudo pip install watchdog

# fix error with jpeg (if you get it)
# uninstall PIL
sudo pip uninstall PIL
# install libjpeg-dev with apt
sudo apt-get install libjpeg-dev
# reinstall pillow
pip install -I pillow
# (from here https://github.com/odoo/odoo/issues/612 )

# fix issue with lessc
# install Less CSS via nodejs according to this instruction:
# https://www.odoo.com/documentation/8.0/setup/install.html

# create postgres user:
sudo su - postgres -c "createuser -s $USER"

# Create new config file if you don't have it yet:
cd /path/to/odoo
./openerp-server --save

# then edit it, e.g. via emacs
emacs -nw ~/.openerp_serverrc

```

(continues on next page)

(continued from previous page)

```

# set dbfilter = ^%h$ 
# set workers = 2 # to make longpolling\bus\im work

# create different versions of conf file:
cp ~/openerp_serverrc ~/openerp_serverrc-9
cp ~/openerp_serverrc ~/openerp_serverrc-8

##### /etc/hosts
# /etc/hosts must contains domains you use, e.g:
sudo bash -c "echo '127.0.0.1 8_0-project1.local' >> /etc/hosts"
sudo bash -c "echo '127.0.0.1 8_0-project2.local' >> /etc/hosts"
sudo bash -c "echo '127.0.0.1 9_0-project1.local' >> /etc/hosts"

##### nginx
# put nginx_odoo.conf to /etc/nginx/sites-enabled/
# delete default configuration:
cd /etc/nginx/sites-enabled/
rm default
# restart nginx
sudo /etc/init.d/nginx restart

##### run Odoo
cd /path/to/odoo
git checkout somebranch-or-revision
git tag 8_0-honduras.local
# everytime run odoo this way:
git checkout 8_0-client1.local && ./odoo.py --config=/path/to/.openerp_serverrc-8
# or
git checkout 8_0-project1.local && ./odoo.py --config=/path/to/.openerp_serverrc-8 --
→auto-reload
# or
git checkout 9_0-project1.local && ./odoo.py --config=/path/to/.openerp_serverrc-9 --
→dev
# etc.
# then open database you need, e.g. (type http:// explicitly, because browser could
→understand it as search request)
# http://8_0-client1.local/
# (database name should be 8_0-client1.local )

```

### 10.1.3 Nginx configuration

Working via nginx is recommended for any type of installation

```

server {
    listen 80 default_server;
    server_name .local;

    proxy_buffers 16 64k;
    proxy_buffer_size 128k;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
    #proxy_redirect http:// https://;
}

```

(continues on next page)

(continued from previous page)

```

proxy_read_timeout 600s;
client_max_body_size 100m;

location /longpolling {
    proxy_pass http://127.0.0.1:8072;
}

location / {
    proxy_pass http://127.0.0.1:8069;
}
}

```

## 10.2 Longpolling

Longpolling is a way to deliver instant notification to web client (e.g. in chats).

To activate longpolling:

- install dependencies

- odoo 11.0

```
python -c "import gevent" || sudo pip3 install gevent
```

- odoo 10.0

```
python -c "import gevent" || sudo pip install gevent
python -c "import psycopgreen" || sudo pip install psycopgreen
```

- set non-zero value for *workers* parameter
- configure nginx

```

location /longpolling {
    proxy_pass http://127.0.0.1:8072;
}
location / {
    proxy_pass http://127.0.0.1:8069;
}

```

- if you install odoo 9.0 via deb package, then you have to restore openerp-gevent file (see #10207):

```
cd /usr/bin/
wget https://raw.githubusercontent.com/odoo/odoo/9.0/openerp-gevent
chmod +x openerp-gevent
```

Read more about longpolling

## 10.3 About longpolling

### What is HTTP Long Polling?

Web applications were originally developed around a client/server model, where the Web client is always the initiator of transactions, requesting data from the server. Thus, there was no mechanism for the server to independently send, or push, data to the client without the client first making a request.

### **In a Nutshell: HTTP Long Polling**

To overcome this deficiency, Web app developers can implement a technique called HTTP long polling, where the client polls the server requesting new information. The server holds the request open until new data is available. Once available, the server responds and sends the new information. When the client receives the new information, it immediately sends another request, and the operation is repeated. This effectively emulates a server push feature.

Thus, each data packet means new connection which will remain open until the server sends the information.

In practice the connection usually reinstalls once per 20-30 seconds to get rid of possible problems (mistakes) , e.g. problems connected with HTTP-proxy.

In contradiction to usual polling, such notice appears faster.

Delay = connection installing + data transfer

### **Advantages of longpolling**

- The loading to the server is reduced unlike usual polling
- Reduced traffic
- Supporting in all modern browsers

Thus, longpolling helps the client to receive data as soon as they appear in the server in contrast to periodic, which send requests according to interval specified.

## **10.4 --workers**

Non-zero values for --workers activates Multiprocessing.

Multiprocessing increases stability, makes somewhat better use of computing resources and can be better monitored and resource-restricted.

- Number of workers should be based on the number of cores in the machine (possibly with some room for cron workers depending on how much cron work is predicted)
- Worker limits can be configured based on the hardware configuration to avoid resources exhaustion

**Warning:** multiprocessing mode currently isn't available on Windows

### **10.4.1 Longpolling**

Hidden feature of Multiprocessing is automatic run gevent process for longpolling support.

Longpolling is an extra process, i.e. if you have --workers=2 then you will get 2 worker processes and 1 gevent process

## **10.5 --db\_maxconn**

Here is definition from `odoo/tools/config.py`

```
group.add_option("--db_maxconn", dest="db_maxconn", type='int', my_default=64,
                help="specify the the maximum number of physical connections to_
                ↪postgresql")
```

More accurate explanation of this option is as following:

`db_maxconn` – specify the the maximum number of physical connections to postgresql **per odoo process, but for all databases**

### How much process odoo runs?

- *longpolling* – no more than 1 process
- *workers*
- *max\_cron\_threads*

### What it means practically?

If you have deployment with big number of databases or simultaneous users you may face following error:

```
2017-09-11 14:01:14,876 8676 ERROR ? odoo.service.server: Worker (8676) Exception_
↪occurred, exiting...
Traceback (most recent call last):
  File "/opt/odoo/10/odoo/service/server.py", line 721, in run
    self.process_work()
  File "/opt/odoo/10/odoo/service/server.py", line 791, in process_work
    db_names = self._db_list()
  File "/opt/odoo/10/odoo/service/server.py", line 784, in _db_list
    db_names = odoo.service.db.list_dbs(True)
  File "/opt/odoo/10/odoo/service/db.py", line 325, in list_dbs
    with closing(db.cursor()) as cr:
  File "/opt/odoo/10/odoo/sql_db.py", line 622, in cursor
    return Cursor(self.__pool, self dbname, self dsn, serialized=serialized)
  File "/opt/odoo/10/odoo/sql_db.py", line 164, in __init__
    self._cnx = pool.borrow(dsn)
  File "/opt/odoo/10/odoo/sql_db.py", line 505, in _locked
    return fun(self, *args, **kwargs)
  File "/opt/odoo/10/odoo/sql_db.py", line 573, in borrow
    **connection_info)
  File "/usr/local/lib/python2.7/dist-packages/psycopg2/__init__.py", line 164, in_
↪connect
    conn = _connect(dsn, connection_factory=connection_factory, async=async)
OperationalError: FATAL:  remaining connection slots are reserved for non-replication_
↪superuser connections
```

To resolve it you need configure following parameters:

- In odoo
  - `db_maxconn`
  - *workers*
  - *max\_cron\_threads*
- In postgresql
  - `max_connections`

Those parameters must satisfy following condition:

```
(1 + workers + max_cron_threads) * db_maxconn < max_connections
```

For example, if you have following values:

- workers = 1 (minimal value to make longpolling work)
- max\_cron\_threads = 2 (default)
- db\_maxconn = 64 (default)
- max\_connections = 100 (default)

then  $(1 + 1 + 2) * 64 = 256 > 100$ , i.e. the condition is not satisfied and such deployment may face the error described above.

## 10.6 --max-cron-threads

Here is definition from `odoo/tools/config.py`

```
group.add_option("--max-cron-threads", dest="max_cron_threads", my_default=2,  
                 help="Maximum number of threads processing concurrently cron jobs",  
                 default=2, type="int")
```

## 10.7 --addons-path

### 10.7.1 Duplicate addons

If you have two folder with the same module and you have reason to add both folders to `addons_path`, then first found version of the module will be used. That is folder in the beginning of `addons_path` list has more priority.

## 10.8 --log-handler

```
--log-handler=PREFIX:LEVEL
```

Setups a handler at LEVEL for a given PREFIX. This option can be repeated.

For example, if you want to have DEBUG level for module `telegram` only, you can run it with parameter:

```
--log-handler=odoo.addons.telegram:DEBUG
```

To disable werkzeug logs add following parameter:

```
--log-handler=werkzeug:CRITICAL
```

To see all odoo log messages:

```
--log-handler=odoo:DEBUG
```

To see all log messages (including ones from libs):

```
--log-handler=:DEBUG
```

### 10.8.1 Log levels

- CRITICAL
- ERROR
- WARNING
- INFO
- DEBUG
- NOTSET

### 10.8.2 Usefull logs

Show api requests:

```
--log-handler=odoo.api:DEBUG
```

## 10.9 --db-filter

The main purpose of `--db-filter` is to avoid asking user which database he needs to use (he may not know it). This is implemented by checking HOST address, which was used.

For example, you have two independent websites, say `shop1.example.com` and `shop2.example.com`, that point to the same odoo server with two databases. By using `--db-filter` you can configure odoo to use corresponding database depending on used host address. Check the documentation links below or jump to examples to find out how to do it.

### 10.9.1 Docs

Official documentation: <https://www.odoo.com/documentation/master/setup/deploy.html#dbfilter>

Core code: [https://github.com/odoo/odoo/search?l=Python&q=%22def+db\\_monodb%22](https://github.com/odoo/odoo/search?l=Python&q=%22def+db_monodb%22)

Additional option: [https://github.com/OCA/server-tools/tree/11.0/dbfilter\\_from\\_header](https://github.com/OCA/server-tools/tree/11.0/dbfilter_from_header)

### 10.9.2 Examples

#### Single database

If you have a single database, you may set default filter:

```
--db-filter=.*
```

## **Ignoring other databases**

To force odoo always use only one database, say mydb, use following filter:

```
--db-filter=^mydb$
```

## **Database names equal to hostname**

```
--db-filter=^%h$
```

To use filter above, you must name databases equal to host address, for example:

- shop1.example.com – name of the first database
- shop2.example.com – name of the second database
- www.super-shop.example.com – name of the third database
- it-projects.info – name of the fourth database

**Warning:** this filter cannot work with and without www prefix at the same time

## **Database names equal to subdomain**

```
--db-filter=^%d$
```

To use filter above, you must name databases equal to subdomain, for example if database name is shop, then the filter will use it for any of following requests:

- shop.example.com
- www.shop.example.com
- shop.yourbrand.example
- www.shop.yourbrand.example

## **10.10 --load**

TODO

## **10.11 PosBox**

Official docs:

- [https://www.odoo.com/documentation/user/9.0/point\\_of\\_sale/overview/setup.html](https://www.odoo.com/documentation/user/9.0/point_of_sale/overview/setup.html)

### 10.11.1 Running PosBox on your computer for development purposes

Running PosBox on your computer is means running the second odoo server instead PosBox.

For run the second odoo server it's necessary to change the configuration settings which is different from the running settings the first odoo server.

For this, just change the `xmlrpc_port` and `longpolling_port` value.

For example, if the run settings for the first odoo server `/path/to/openerp-server1.conf`:

```
xmlrpc_port = 8069
longpolling_port = 8072
```

then the settings for the second odoo server `/path/to/openerp-server2.conf` can be as follows:

```
xmlrpc_port = 9069
longpolling_port = 9072
```

Example of running **PosBox** on your computer with used Network Printer:

- Run first Odoo Server, e.g.:

```
./openerp-server --config=/path/to/openerp-server1.conf
```

- Install the **Pos Printer Network** module on Odoo in a usual way.
- Configure PosBox using the [installation instructions](#).
- Run second Odoo Server using new settings and add to `--load` parameters, e.g.:

```
./openerp-server --load=web,hw_proxy,hw_posbox_homepage,hw_scale,hw_scanner,hw_
↳escpos,hw_printer_network --config=/path/to/openerp-server2.conf
```

- Print in network printer.

### Run PosBox via docker

Example with `hw_printer_network` and **PosBox 8.0**:

```
docker run -d -p 1984:1984 --name wdb kozea/wdb
docker run -d -e POSTGRES_USER=odoo -e POSTGRES_PASSWORD=odoo --name db-posbox-8.0 \
↳postgres:9.5

docker run \
-p 9069:8069 \
-p 9072:8072 \
--link wdb:wdb -e WDB_SOCKET_SERVER=wdb -e WDB_NO_BROWSER_AUTO_OPEN=True \
-e OODO_MASTER_PASS=admin \
--privileged \
-v /dev/bus/usb:/dev/bus/usb \
--name 8.0-posbox \
--link db-posbox-8.0:db \
-t itprojectsllc/install-odoo:8.0-posbox -- --load=web,hw_proxy,hw_posbox_homepage,
↳hw_scale,hw_scanner,hw_escpos,hw_printer_network
```

To use your version of built-in odoo modules use add following `-v path/to/odoo:/mnt/odoo-source`.

Source of this docker can be found here: <https://github.com/it-projects-llc/install-odoo/tree/8.0/dockers/posbox>

**Warning:** It actually doesn't work and raises error "No backend available". Probably -device=/dev/SOMETHING has to be used instead of "-privileged -v /dev/bus/usb:/dev/bus/usb"

### 10.11.2 PosBox installation

Download last version posbox\_image:

- <https://nightly.odoo.com/master/posbox/>

---

**Note:** Use another computer with an SD card reader to install the image.

---

You will need to use an image writing tool to install the image you have downloaded on your SD card.

**Etcher** is a graphical SD card writing tool that works on Mac OS, Linux and Windows, and is the easiest option for most users. Etcher also supports writing images directly from the zip file, without any unzipping required. To write your image with Etcher:

- Download [Etcher](#) and install it.
- Connect an SD card reader with the SD card inside.
- Open Etcher and select from your hard drive the Raspberry Pi .img or .zip file you wish to write to the SD card.
- Select the SD card you wish to write your image to.
- Review your selections and click 'Flash!' to begin writing data to the SD card.

#### Connect peripheral devices

Officially supported hardware is listed on [the POS Hardware page](#), but other hardware might work as well.

- **Printer:** Connect an ESC/POS printer to a USB port and power it on.
- **Cash drawer:** The cash drawer should be connected to the printer with an RJ25 cable.
- **Barcode scanner:** Connect your barcode scanner. In order for your barcode scanner to be compatible it must behave as a keyboard and must be configured in US QWERTY. It also must end barcodes with an Enter character (keycode 28). This is most likely the default configuration of your barcode scanner.
- **Scale:** Connect your scale and power it on.
- **Ethernet:** If you do not wish to use Wi-Fi, plug in the Ethernet cable. Make sure this will connect the POSBox to the same network as your POS device.
- **Wi-Fi:** If you do not wish to use Ethernet, plug in a Linux compatible USB Wi-Fi adapter. Most commercially available Wi-Fi adapters are Linux compatible. Officially supported are Wi-Fi adapters with a Ralink 5370 chipset. Make sure not to plug in an Ethernet cable, because all Wi-Fi functionality will be bypassed when a wired network connection is available.
- **Network Printer:** Connect Network Printer.

#### Power the POSBox

Plug the power adapter into the POSBox, a bright red status led should light up.

#### Make sure the POSBox is ready

Once powered, The POSBox needs a while to boot. Once the POSBox is ready, it should print a status receipt with its IP address. Also the status LED, just next to the red power LED, should be permanently lit green.

More information read the:

- <https://www.raspberrypi.org/documentation/installation/installing-images/>
- [https://www.odoo.com/documentation/user/9.0/point\\_of\\_sale/overview/setup.html](https://www.odoo.com/documentation/user/9.0/point_of_sale/overview/setup.html)

### 10.11.3 Introduction

The **POSBox** runs a heavily modified **Raspbian Linux** installation, a Debian derivative for the **Raspberry Pi**. It also runs a barebones installation of Odoo which provides the webserver and the drivers. The hardware drivers are implemented as Odoo modules. Those modules are all prefixed with `hw_*` and they are the only modules that are running on the POSBox. Odoo is only used for the framework it provides. No business data is processed or stored on the POSBox. The Odoo instance is a shallow git clone of the `8.0` branch.

The root partition on the POSBox is mounted `read-only`, ensuring that we don't wear out the SD card by writing to it too much. It also ensures that the filesystem cannot be corrupted by cutting the power to the POSBox. Linux applications expect to be able to write to certain directories though. So we provide a ramdisk for `/etc` and `/var` (Raspbian automatically provides one for `/tmp`). These ramdisks are setup by `setup_ramdisks.sh`, which we run before all other init scripts by running it in `/etc/init.d/rcS`. The ramdisks are named `/etc_ram` and `/var_ram` respectively. Most data from `/etc` and `/var` is copied to these tmpfs ramdisks. In order to restrict the size of the ramdisks, we do not copy over certain things to them (eg. apt related data). We then bind mount them over the original directories. So when an application writes to `/etc/foo/bar` it's actually writing to `/etc_ram/foo/bar`. We also bind mount `/` to `/root_bypass_ramdisks` to be able to get to the real `/etc` and `/var` during development.

### 10.11.4 How to edit config

If you have the POSBox's IP address and an SSH client you can access the POSBox's system remotely.

**Login:** pi **Password:** raspberry

Beware that root `(/)` is mounted read only and so you cannot use write.

If you want to use it you need to reboot in normal mode.

```
sudo su
mount -o rw,remount /
mount -o rw,remount /root_bypass_ramdisks
```

sync and reboot posbox

```
sync
reboot
```

### 10.11.5 How to update odoo command-line options

edit `/root_bypass_ramdisks/etc/init.d/odoo`

```
nano /root_bypass_ramdisks/etc/init.d/odoo
```

add `hw_printer_network` to `--load` parameter

```
$LOGFILE --load=web,hw_proxy,hw_posbox_homepage,hw_posbox_upgrade,hw_scale,hw_scanner,
↪hw_escpos,hw_blackbox_be,hw_screen,hw_printer_network
```

### 10.11.6 How to edit odoo source

Comment out line 354 in `hw_escpos/controllers/main.py`

```
nano /home/pi/odoo/addons/hw_escpos/controllers/main.py
```

i.e. replace `driver.push_task('printstatus')` with

```
# driver.push_task('printstatus')
```

sync and reboot posbox

```
sync  
reboot
```

### 10.11.7 Reading logs on posbox

Reading logs

```
tail -f /var/log/odoo/odoo-server.log
```

Edit log level:

```
nano /home/pi/odoo/addons/point_of_sale/tools/posbox/configuration/odoo.conf
```

replace to

```
log_level = info
```

# CHAPTER 11

---

## Continuous Delivery

---

TODO



# CHAPTER 12

---

## Maintenance

---

### 12.1 Data Migration

Data Migration is a process of keeping correct data in database after updating to new module version. For example, simple field renaming leads to data lost if you don't have proper data migration scripts.

For *Module Migration* see [Porting Modules](#)

#### 12.1.1 Preparing

Those migrations are between module version.

From Odoo <https://github.com/odoo/odoo/blob/11.0/odoo/modules/migration.py#L53>:

This class manage the migration of modules. Migrations files must be python files containing a `migrate(cr, installed_version)` function. Theses files must respect a directory tree structure: A 'migrations' folder which contains a folder by version. Version can be 'module' version or 'server.module' version (in this case, the files will only be processed by this version of the server). Python file names must start by `pre` or `post` and will be executed, respectively, before and after the module initialisation. `end` scripts are run after all modules have been updated.

Example:

```
<moduledir>
`-- migrations
    |-- 1.0
    |   |-- pre-update_table_x.py
    |   |-- pre-update_table_y.py
    |   |-- post-create_plop_records.py
    |   |-- end-cleanup.py
    |   `-- README.txt                  # not processed
    |-- 9.0.1.1                         # processed only on a 9.0 server
    |   |-- pre-delete_table_z.py
    |   `-- post-clean-data.py
    '-- foo.py                            # not processed
```

### 12.1.2 Execution

Migration files are just code files that don't need to be registered anywhere. When updating an addon Odoo searching in the *migrations* for folders with a version in between, up to, and including the version that is in updating for. It happens before all other files were observed, so at this moment nothing is changed at your database layout. Then, if folders are found Odoo executes python files with prefix *pre-* in it. They should contain a defined function called *migrate*. This function has two arguments: database cursor and currently installed version.

After all pre-migrate functions were successfully executed, Odoo updates the module. Now, the database might be different from the previous version one. For example, if in a new version we changed the model field type, in the database this column will be changed without data preserving. Or if a field was renamed, in the new version just a new column will be created.

Then, after the module was updated, Odoo search for post-migrate files by the same algorithm and execute them.

*end* scripts are run after all modules have been updated.

**Warning:** Migration updates are not rollbacks if some errors happened later during modules updating process. So, you shall always try to update module with migration scripts on a copy first.

### 12.1.3 Example

*POS Debt & Credit notebook*. We need to preserve credit\_product field data in the product.template model after updating to a newer version. In previous version it was boolean field, now it is a many2one field with the relation to account.journal model. Here, we, using a temporary column, calculate transfer data from boolean to many2one credit\_product field.

*pre-migrate.py*:

```
def migrate(cr, version):
    # Add temporary credit product column
    cr.execute('ALTER TABLE product_template ADD temporary_credit_product int')
    # Select debt journals
    cr.execute('SELECT id FROM account_journal WHERE account_journal.debt is true')
    # Take the first journal
    journal_id = cr.fetchone()
    if journal_id:
        # set token one to all credit products
        cr.execute('UPDATE product_template SET temporary_credit_product=%s WHERE
        ↴credit_product is true', journal_id)
```

*post-migrate.py*:

```
def migrate(cr, version):
    # update new credit_product column with the temporary one
    cr.execute('UPDATE product_template SET credit_product=temporary_credit_product')
    # Drop temporary column
    cr.execute('ALTER TABLE product_template DROP COLUMN temporary_credit_product')
```

# CHAPTER 13

---

IDE

---

## 13.1 Emacs

### 13.1.1 Emacs

Install emacs 24.4+ <http://askubuntu.com/questions/437255/how-to-install-emacs-24-4-on-ubuntu>

- Open Emacs
- Press Alt-x package-list-packages
- install packages: click i and then x
- some packages require dependencies, that have to be installed via terminal \* flymake \* loccur \* flymake-css \* flymake-jslint \* flymake-python-pyflakes

```
sudo pip install flake8
```

- magit
- js3-mode

### 13.1.2 Spacemacs

#### Requirements

- emacs version 24 or newer.

#### Installation

Install spacemacs from github <https://github.com/syl20bnr/spacemacs>

## Documentation

<http://spacemacs.org/doc/DOCUMENTATION.html>

### Layers for Odoo development

Use the following layers:

- auto-completion
- better-defaults
- emacs-lisp
- git
- syntax-checking
- version-control
- python
- eyebrowse
- sql
- python
- semantic

Syntax-checking in python uses pylint package ([http://liuluheng.github.io/wiki/public\\_html/Python/flycheck-pylint-emacs-with-python.html](http://liuluheng.github.io/wiki/public_html/Python/flycheck-pylint-emacs-with-python.html)). Install it by

```
sudo pip install pylint
```

### 13.1.3 Replace text in recursively found files

1. M-x find-name-dired: you will be prompted for a root directory and a filename pattern.
2. Press t to “toggle mark” for all files found.
3. Press Q for “Query-Replace in Files...”: you will be prompted for query/substitution regexps.
4. Proceed as with query-replace-regexp: SPACE to replace and move to next match, n to skip a match, etc.

Based on: <http://stackoverflow.com/questions/270930/using-emacs-to-recursively-find-and-replace-in-text-files-not-already-open>

### 13.1.4 Pylint

Pylint is a tool that checks for errors in Python code, tries to enforce a coding standard and looks for code smells. It can also look for certain type errors, it can recommend suggestions about how particular blocks can be refactored and can offer you details about the code’s complexity. <https://pylint.readthedocs.io/en/latest/>

Install pylint.

```
sudo pip install pylint
```

With the Flycheck emacs extension, pylint’s output will be shown right inside your emacs buffers. Spacemacs has flycheck in his syntax-checking layer.

```
M-x package-install RET flycheck
```

Configure pylint by using a pylintrc file.

```
pylint --generate-rcfile >.pylintrc
```

## Pylint Odoo plugin

Install pylint odoo plugin <https://github.com/OCA/pylint-odoo>

```
pip install --upgrade git+https://github.com/oca/pylint-odoo.git
```

or

```
pip install --upgrade --pre pylint-odoo
```

Add the plugin in pylintrc.

```
load-plugins=pylint_odoo
```

## Useful configurations

By default there is 100 characters per line allowed. Allow 120 characters

```
max-line-length=120
```

To disable certain warning add its code to disable list in pylintrc. For example, If you don't like this message Missing method docstring with code C0111 or this Use of super on an old style class (E1002)

```
disable=E1608,W1627,E1601,E1603,E1602,E1605,E1604,E1607,E1606,W1621,W1620,W1623,W1622,  
˓→W1625,W1624,W1609,W1608,W1607,W1606,W1605,W1604,W1603,W1602,W1601,W1639,W1640,I0021,  
˓→W1638,I0020,W1618,W1619,W1630,W1626,W1637,W1634,W1635,W1610,W1611,W1612,W1613,W1614,  
˓→W1615,W1616,W1617,W1632,W1633,W0704,W1628,W1629,W1636,C0111,E1002
```

You can find other codes here: <http://pylint-messages.wikidot.com/>

Flychek highlights odoo import lines as from openerp import models, fields, api with error message F0401: Unable to import.... There are two options to fix it - <http://stackoverflow.com/questions/1899436/pylint-unable-to-import-error-how-to-set-pythonpath>.

Edit pylintrc to include your odoo directory like this:

```
init-hook='import sys; sys.path.append("/path/to/odoo")'
```

## 13.2 PyCharm

### 13.2.1 PyCharm

#### Remote access with pgAdmin to Odoo postgres database on Ubuntu

This is for PgAdmin integration, but same method working with PyCharm.

STEP #1 – get pgAdmin Install pgAdmin from pgadmin.org

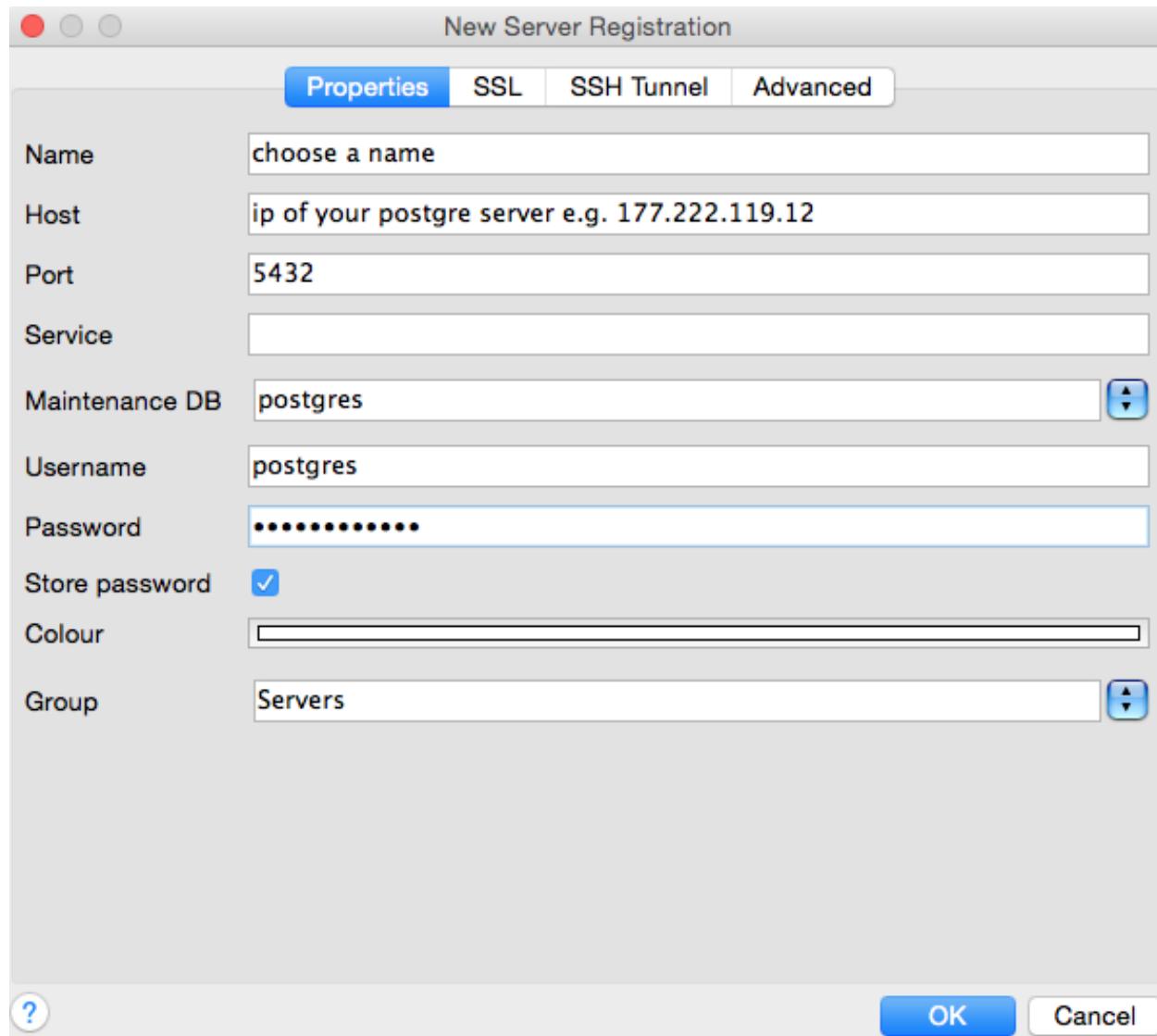
STEP #2 – allow postgre server remote connections from everywhere Open etc/postgresql/9.x/main/pg\_hba.conf and add following line: host all all all md5

STEP #3 – let the postgre server listen to everyone Open etc/postgresql/9.x/main/postgresql.conf and change following line: listen\_addresses = '\*'

STEP #4 – give the user “postgres” a password Start the psql terminal: sudo -u postgres psql Give a password: ALTER USER postgres PASSWORD ‘yourpassword’; Leave the psql terminal: q

STEP #5 Restart postgre server by executing this terminal command: sudo /etc/init.d/postgresql restart

STEP #6 Start pgAdmin and add a connection to a server like this:



You are ready!

Original:

<http://odoo.guide/remote-access-with-pgadmin-to-odoo-postgre-database-on-ubuntu/>

## 13.3 Tmux

### 13.3.1 Tmux installation

#### Install Tmux

```
sudo apt-get install tmux
```

Check version

```
tmux -V
```

If you have 1.8 or older then you should update. Here are update commands for ubuntu 14.04

```
sudo apt-get update
sudo apt-get install -y python-software-properties software-properties-common
sudo add-apt-repository -y ppa:pi-rho/dev
sudo apt-get update
sudo apt-get install -y tmux=2.0-1~ppa1~t
```

Now if you do tmux -V it should show tmux 2.0 which is a good version for tmux plugins.

Based on: <http://stackoverflow.com/questions/25940944/upgrade-tmux-from-1-8-to-1-9-on-ubuntu-14-04>

#### Install Tmux Plugin Manager

Requirements: tmux version 1.9 (or higher), git, bash

Clone TPM:

```
$ git clone https://github.com/tmux-plugins/tpm ~/.tmux/plugins/tpm
```

Put this at the bottom of .tmux.conf:

```
# List of plugins
set -g @plugin 'tmux-plugins/tpm'
set -g @plugin 'tmux-plugins/tmux-sensible'

# Other examples:
# set -g @plugin 'github_username/plugin_name'
# set -g @plugin 'git@github.com:user/plugin'
# set -g @plugin 'git@bitbucket.com/user/plugin'

# Initialize TMUX plugin manager (keep this line at the very bottom of tmux.conf)
run '~/.tmux/plugins/tpm/tpm'
```

Reload TMUX environment so TPM is sourced:

```
# type this in terminal
$ tmux source ~/.tmux.conf
```

Based on: <https://github.com/tmux-plugins/tpm>

## Install Tmux Resurrect

Add plugin to the list of TPM plugins in .tmux.conf:

```
set -g @plugin 'tmux-plugins/tmux-resurrect'
```

Hit prefix + I to fetch the plugin and source it. You should now be able to use the plugin.

Based on: <https://github.com/tmux-plugins/tmux-resurrect>

## Install tmux-continuum

Last saved environment is automatically restored when tmux is started. Put the following lines in tmux.conf:

```
set -g @continuum-save-interval '5'  
set -g @continuum-restore 'on'
```

Your environment will be automatically saved every 5 minutes. When you start tmux it will automatically restore

Based on: <https://github.com/tmux-plugins/tmux-continuum>

### 13.3.2 Tmux configuration

Create a file with the name .tmux.conf in your home directory.

An example of .tmux.conf:

```
# Global settings  
  
# Set prefix key to Ctrl-a  
# unbind-key C-b  
# set-option -g prefix C-a  
  
# send the prefix to client inside window  
# bind-key C-a send-prefix  
  
# scrollback buffer n lines  
set -g history-limit 10000  
  
# tell tmux to use 256 colour terminal  
set -g default-terminal "screen-256color"  
  
# enable wm window titles  
set -g set-titles on  
  
# reload settings  
bind-key R source-file ~/.tmux.conf  
  
# Statusbar settings  
  
# toggle statusbar  
bind-key s set status  
  
# use vi-style key bindings in the status line  
set -g status-keys vi
```

(continues on next page)

(continued from previous page)

```

# amount of time for which status line messages and other indicators
# are displayed. time is in milliseconds.
set -g display-time 2000

# default statusbar colors
set -g status-fg white
set -g status-bg default
set -g status-attr default

# default window title colors
setw -g window-status-fg white
setw -g window-status-bg default
setw -g window-status-attr dim

# active window title colors
setw -g window-status-current-fg cyan
setw -g window-status-current-bg default
#setw -g window-status-current-attr bright
setw -g window-status-current-attr underscore

# command/message line colors
set -g message-fg white
set -g message-bg black
set -g message-attr bright

set-option -g status-keys vi
set-option -g mode-keys vi

# List of plugins
set -g @plugin 'tmux-plugins/tpm'
set -g @plugin 'tmux-plugins/tmux-sensible'
set -g @plugin 'tmux-plugins/tmux-resurrect'
set -g @plugin 'tmux-plugins/tmux-continuum'
set -g @continuum-save-interval '5'
set -g @continuum-restore 'on'

# Other examples:
# set -g @plugin 'github_username/plugin_name'
# set -g @plugin 'git@github.com/user/plugin'
# set -g @plugin 'git@bitbucket.com/user/plugin'

# Initialize TMUX plugin manager (keep this line at the very bottom of tmux.conf)
run '~/.tmux/plugins/tpm/tpm'

```

## 13.4 Visual Studio Code

### 13.4.1 Install Visual Studio Code

- install visualstudiocode from <https://code.visualstudio.com>
- **add the following Extensions :**
  - python: <https://marketplace.visualstudio.com/items?itemName=donjayamanne.python>
  - odoo-snippets: <https://marketplace.visualstudio.com/items?itemName=jeffery9.odoo-snippets>

- Follow the same instructions in (emacs-pylint) to install pylint and Pylint Odoo plugin. Then make same configurations in pylintrc file as described there.

**Attention:** pylintrc file can be placed in the user environment to work for all projects. like for debian “`~/.pylintrc`”

### 13.4.2 Configuration:-

#### sample configuration (for user or workspace setting)

```
// Place your settings in this file to overwrite default and user settings.
{
    // "python.pythonPath": "optional: path to python use if you have environment path
    //",

    // use this so the autocomplete/goto definition will work with python extension
    "python.autoComplete.extraPaths": [
        "${workspaceRoot}/odoo addons",
        "${workspaceRoot}/odoo",
        "${workspaceRoot}/odoo/openerp addons" ],

    // "python.linting.pylintPath": "optional: path to python use if you have
    // environment path",

    "python.linting.enabled": true,

    //load the pylint_odoo

    "python.linting.pylintArgs": ["--load-plugins", "pylint_odoo"],

    "python.formatting.provider": "yapf",

    // "python.formatting.yapfPath": "optional: path to python use if you have
    // environment path",

    // "python.linting.pep8Path": "optional: path to python use if you have
    // environment path",

    "python.linting.pep8Enabled": true,

    // add this auto-save option so the pylint will show errors while editing otherwise
    // it will only show the errors on file save
    "files.autoSave": "afterDelay",
    "files.autoSaveDelay": 500

    // The following will hide the compiled file in the editor/ add other file to
    // hide them from editor
    "files.exclude": {
        "**/*.pyc": true
    }
}
```

---

**Note:** some lines are commented because it is optional. you can activate them if needed like in the case of using Virtualenv.

---

### 13.4.3 Debugging

#### Launch Configurations

To debug your app in VS Code, you'll first need to set up your launch configuration file - `launch.json`. Click on the Configure gear icon on the Debug view top bar, choose your debug environment and VS Code will generate a `launch.json` file under your workspace's `.vscode` folder.

#### sample python Debugging

```
{
    "name": "Python",
    "type": "python",
    "request": "launch",
    "stopOnEntry": false,
    "pythonPath": "${config.python.pythonPath}",
    // "program": "${file}", use this to debug opened file.
    "program": "${workspaceRoot}/Path/To/odoo.py",
    "args": [
        "-c ${workspaceRoot}/sampleconfigurationfile.cfg"
    ],
    "cwd": "${workspaceRoot}",
    "console": "externalTerminal",
    "debugOptions": [
        "WaitOnAbnormalExit",
        "WaitOnNormalExit",
        "RedirectOutput"
    ]
},
```

---

**Important:** use “args” to specify any options like database, config or user name and password.

---

sorce



# CHAPTER 14

---

## Remote Development

---

The section contains instructions to setup remote development environment. That is developer runs odoo and probably other tools on remote server rather on his machine. Advantages of this approach are:

- easy way to provide big computing capacity
- the same environment from any device
- easy way to demonstrate work

### 14.1 Usage

#### 14.1.1 SSH agent forwarding

To send commit or get access to private repositories you can use either login-password authentication or ssh keys. In later case you can face a problem to do it on remote server, because your private ssh key is not installed there. The good news is that you don't need to do it. You can "forward ssh keys". Just add `-A` to your ssh command or add following lines to your ssh config (`~/.ssh/config`) on your (local) computer:

```
Host your.dev.server.example.com
  ForwardAgent yes
```

Then connect to your server and type to test:

```
ssh -T git@github.com
```

For more information see: <https://developer.github.com/guides/using-ssh-agent-forwarding/>

#### Putty users (Windows)

- install Pageant SSH agent (`pageant.exe`) <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>
- add your keys to Pageant SSH

- enable ssh agent forwarding in putty settings

### 14.1.2 How to mount local files on a server

#### sshfs

On your local machine:

```
# Step 1. Install ssh server on your local machine
# TODO
# Step 2. Configure ssh keys on you local machine
cat cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
# Step 3. Connect to your server
ssh USERNAME@SERVER -p 22 -A -R 10000:localhost:22
```

On your remote server:

```
# Step 4. Mount your directory on remote server
sshfs -p 10000 -o idmap=user,nonempty \
      LOCALUSERNAME@127.0.0.1:/PATH/TO/LOCAL/FOLDER /PATH/TO/REMOTE/FOLDER
```

#### References

- <https://superuser.com/questions/616182/how-to-mount-local-directory-to-remote-like-sshfs>

### 14.1.3 Remote desktop via X2GO

#### Deploying X2GO server

x2go allows you to run remotely browser (or any other application on x-server)

- Connect to your server:
- install x2go server :

```
sudo add-apt-repository ppa:x2go/stable && \
sudo apt-get update && \
sudo apt-get install -y x2goserver x2goserver-xsession
```

- install desktop environment you prefer, e.g. LXDE:

```
sudo apt-get install lubuntu-desktop
# choose lightdm
```

- Install browser Pale Moon

```
# http://linux.palemoon.org
sudo sh -c "echo 'deb http://download.opensuse.org/repositories/home:/stevenpusser/
↳xUbuntu_18.04/ /' > /etc/apt/sources.list.d/home:stevenpusser.list" && \
sudo apt-get update && \
sudo apt-get install palemoon
```

## X2GO Client

- install x2goclient

Ubuntu:

```
sudo add-apt-repository ppa:x2go/stable && \
sudo apt-get update && \
sudo apt-get install x2goclient
```

References:

- <https://www.howtoforge.com/tutorial/x2go-server-ubuntu-14-04/>
- <http://wiki.x2go.org/doku.php/doc:installation:x2goclient>

- Run client:

```
x2goclient
```

- create a new session with the settings below and connect to it (we assume that you have user named “noroot” with ssh keys configured):

```
Host : YOHOST
Port : 22
Session type: LXDE
[x] Try auto Login
Input / Output: Use Whole Display
Username: noroot
```

## 14.2 Containers administration

### 14.2.1 LXD Containers

```
# For understanding LXC see https://wiki.debian.org/LXC

# Based on:
# lxd + docker: https://stgraber.org/2016/04/13/lxd-2-0-docker-in-lxd-712/
# lxd network (static ip): https://stgraber.org/2016/10/27/network-management-with-lxd-2-3/
LXD_NETWORK="dev-network2"

# install lxd 2.3+
apt-get install software-properties-common iptables-persistent

add-apt-repository ppa:ubuntu-lxc/lxd-stable
apt-get update
apt-get dist-upgrade
apt-get install lxd

# init lxd
lxd init

# init network
lxc network create ${LXD_NETWORK}
lxc network show ${LXD_NETWORK} # check ipv4.address field
```

(continues on next page)

(continued from previous page)

```
#####
# Per each Developer
GITHUB_USERNAME="yelizariev"
CONTAINER="${GITHUB_USERNAME}"
SERVER_DOMAIN="${GITHUB_USERNAME}.dev.it-projects.info"
NGINX_CONF="dev-${GITHUB_USERNAME}.conf"
LOCAL_IP="10.37.82.100" # use one from network subnet
PORT="10100" # unique per each developer

# https://discuss.linuxcontainers.org/t/docker-cannot-write-to-devices-allow/998/3
read -r -d '' RAW_LXC <<EOF
lxc.apparmor.profile=unconfined
lxc.mount.auto="proc:rw sys:rw cgroup:rw"
lxc.cgroup.devices.allow=a
lxc.cap.drop=
EOF
lxc init ubuntu-daily:18.04 ${CONTAINER} -p default && \
lxc network attach ${LXD_NETWORK} ${CONTAINER} eth0 && \
lxc config device set ${CONTAINER} eth0 ipv4.address ${LOCAL_IP} && \
lxc config set ${CONTAINER} security.privileged true && \
# allow run docker in privileged mode.
# https://discuss.linuxcontainers.org/t/failed-to-write-a-rwm-to-devices-allow-
#→operation-not-permitted-in-privileged-container/925/3
lxc config set ${CONTAINER} raw.lxc "$RAW_LXC"

# forward ssh port
iptables -t nat -A PREROUTING -p tcp --dport ${PORT} -j DNAT \
--to-destination ${LOCAL_IP}:22

# save iptables record. Otherwise it's disappeared after rebooting
sudo netfilter-persistent save
sudo netfilter-persistent reload

PASS=$(< /dev/urandom tr -dc _A-Za-z0-9 | head -c${1:-32};echo;)"
lxc start ${CONTAINER}

lxc exec ${CONTAINER} -- apt-get update && \
lxc exec ${CONTAINER} -- apt dist-upgrade -y

# colorize prompt:
lxc exec ${CONTAINER} -- sed -i "s/#force_color_prompt=yes/force_color_prompt=yes/" /root/.bashrc && \
lxc exec ${CONTAINER} -- sed -i "s/01;32m/01;36m/" /root/.bashrc && \
# install some packages
lxc exec ${CONTAINER} -- apt install docker.io htop python3-pip -y && \
lxc exec ${CONTAINER} -- ln -s /usr/bin/pip3 /usr/bin/pip && \
lxc exec ${CONTAINER} -- pip install odooup && \
# https://docs.docker.com/v17.09/compose/install/#install-compose
lxc exec ${CONTAINER} -- curl -L https://github.com/docker/compose/releases/download/`uname -s`-`uname -m` -o /usr/local/bin/docker-compose && \
lxc exec ${CONTAINER} -- chmod +x /usr/local/bin/docker-compose && \
# update git. See https://github.com/xoe-labs/odooup/issues/8
# TODO: this may not be needed in ubuntu 18
lxc exec ${CONTAINER} -- add-apt-repository ppa:git-core/ppa -y && \
```

(continues on next page)

(continued from previous page)

```

lxc exec ${CONTAINER} -- apt-get update && \
lxc exec ${CONTAINER} -- apt-get install git -y && \
lxc exec ${CONTAINER} -- adduser noroot --disabled-password --gecos "" && \
lxc exec ${CONTAINER} -- mkdir -p /root/.ssh && \
lxc exec ${CONTAINER} -- bash -c "curl --silent https://github.com/${GITHUB_USERNAME}.\n→keys >> /root/.ssh/authorized_keys" && \
# access for noroot
lxc exec ${CONTAINER} -- bash -c "echo $PASS > /root/noroot-password" && \
lxc exec ${CONTAINER} -- bash -c "echo noroot:$PASS | chpasswd" && \
lxc exec ${CONTAINER} -- sudo -u "noroot" bash -c "mkdir -p /home/noroot/.ssh" && \
lxc exec ${CONTAINER} -- sudo -u "noroot" bash -c "curl --silent https://github.com/$\n→{GITHUB_USERNAME}.keys >> /home/noroot/.ssh/authorized_keys" && \
lxc exec ${CONTAINER} -- sudo -u "noroot" sed -i "s/01;32m/01;93m/" /home/noroot/.\n→bashrc && \
# Manage Docker as a non-root user https://docs.docker.com/install/linux/linux-\n→postinstall/
lxc exec ${CONTAINER} -- usermod -aG docker noroot && \
lxc exec ${CONTAINER} -- usermod -aG sudo noroot && \
lxc exec ${CONTAINER} -- locale-gen --purge en_US.UTF-8 && \
lxc exec ${CONTAINER} -- bash -c "echo -e 'LANG=\"en_US.UTF-8\"\nLANGUAGE=\"en_US:en\"\n→\"\\n\" > /etc/default/locale"

lxc config device add ${CONTAINER} sharedcachenoroot disk path=/home/noroot/.cache_\n→source=/var/lxc/share/cache && \
lxc stop ${CONTAINER} && \
lxc start ${CONTAINER}

## nginx on host machine
cd /tmp/
curl -s https://raw.githubusercontent.com/it-projects-llc/odoo-development/master/\n→docs/remote-dev/lxd/nginx.conf > nginx.conf
sed -i "s/NGINX_SERVER_DOMAIN/.${SERVER_DOMAIN}/g" nginx.conf
sed -i "s/SERVER_HOST/${LOCAL_IP}/g" nginx.conf
cp nginx.conf /etc/nginx/sites-available/${NGINX_CONF}
ln -s /etc/nginx/sites-available/${NGINX_CONF} /etc/nginx/sites-enabled/${NGINX_CONF}
# then restart nginx in a usual way

#####
# Control commands

# delete container
lxc delete CONTAINER-NAME

# see iptables rules
iptables -L -t nat

# delete nat rule
iptables -t nat -D PREROUTING POSITION_NUMBER

```



# CHAPTER 15

---

Other

---

## 15.1 RST format

### 15.1.1 Document Title / Subtitle

The title of the whole document is distinct from section titles and may be formatted somewhat differently (e.g. the HTML writer by default shows it as a centered heading).

To indicate the document title in reStructuredText, use a unique adornment style at the beginning of the document. To indicate the document subtitle, use another unique adornment style immediately after the document title. For example:

```
=====
Document Title
=====

-----
Subtitle
-----

Section Title
=====

...
```

Note that “Document Title” and “Section Title” above both use equals signs, but are distinct and unrelated styles. The text of overline-and-underlined titles (but not underlined-only) may be inset for aesthetics.

### Sections

- # with overline, for parts
- \* with overline, for chapters
- =, for sections

- -, for subsections
- ^, for subsubsections
- “, for paragraphs

#### Code block

Enter double colon (::) and then empty line and then at least one space and finally you can enter your code.

Also you can use `inplace code reference` by using ““.

### 15.1.2 Selection

- **bold**
- **italic**
- ``code``

### 15.1.3 Lists

- \* - not numerated
- #. - numerated
- 1,2,3, . . . - numerated

### 15.1.4 Links

- internal link:

```
:doc:`Link Text<.../relative/path/to/article>`
```

- external link:

```
`Link Text <https://google.com>`_
```

### 15.1.5 More documentations

- <http://docutils.sourceforge.net/docs/user/rst/quickref.html>

## 15.2 Adjust chromium window size script

You can make screenshot with size exactly you need.

Open chromium. Do not expand window (or it won't work). Run this command:

```
wmctrl -a chromium -e 1,0,0,760,451
```

Last two arguments is width and height. Consider to add chromium window borders to your screenshot size. In my case it 10px to width and 80px to height. Likely you got same. So for 750 x 371 it be 760 x 451.

---

### Need our service?

For module development contact us by [email](#) or fill out [request form](#):

- [it@it-projects.info](mailto:it@it-projects.info)
  - <https://www.it-projects.info/page/website.contactus>
-