Odoo development Documentation

Release master

IT-Projects LLC

Oct 26, 2017

Contents

- Ask new questions: https://github.com/it-projects-llc/odoo-development/issues/new
- Check open questions: https://github.com/it-projects-llc/odoo-development/issues
- Push your answers and improvements: https://github.com/it-projects-llc/odoo-development

Current content:

CHAPTER 1

First steps

- *Install odoo*
- take the course Bulding 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 you machine, make commits, push updates, create Pull Request

CHAPTER 2

Module Development

Docs and manifests

Files

All files from this section ought to be fully*[0] prepared **before** any other files in new module. It helps you to review requirements again before you start.

README.rst

[0] The only exception could be made for lists of files in `__manifest__.py` (*"data"*, *"qweb"*, *"demo"* fields).

Guidlines

```
===============
 {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>`__

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>`_

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/l10n-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 03bc8c5f9ac53a3349c1caac222f7619a632ccd8
```

commit sha can be found as following

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

doc/index.rst

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

Installation
============

* `Install <https://odoo-development.readthedocs.io/en/latest/odoo/usage/install-
↪module.html>`__ this module in a usual way
```

```
* {OPTIONAL }`Activate longpolling <https://odoo-development.readthedocs.io/en/latest/
↪admin/longpolling.html>`__
* {Additional notes if any}

Configuration
=============

{Instruction how to configure the module.}

* `Activate Developer Mode <https://odoo-development.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/

__manifest__.py (__openerp__.py)

- *Filename*
- *Template*
- *name*
- *summary*
- *category*
  - *Hidden*
- *version*
  - *version in OCA*
- *author*
  - *author in OCA*
- *website*
- *license*
- *external_dependencies*

Filename

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

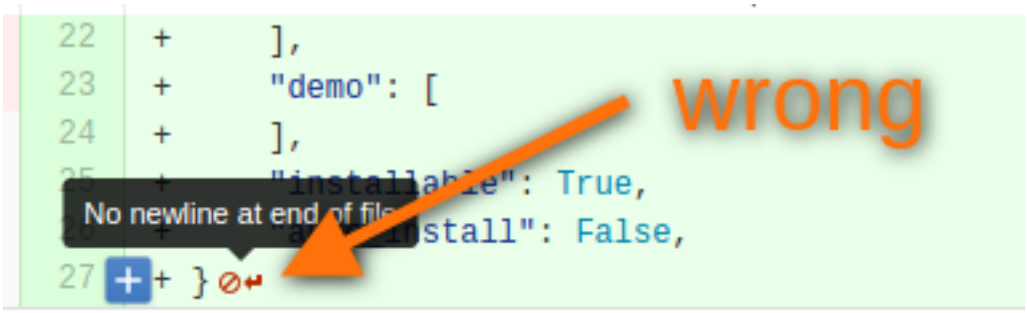
Template

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 })

```
# -*- coding: utf-8 -*-
{
    "name": """{MODULE_NAME}""",
    "summary": """{SHORT_DESCRIPTION_OF_THE_MODULE}""",
    "category": "{MODULE_CATEGORY}",
    # "live_test_url": "",
    "images": [],
    "version": "1.0.0",
    "application": False,

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

    "depends": [
        "{DEPENDENCY1}",
        "{DEPENDENCY2}",
    ],
    "external_dependencies": {"python": [], "bin": []},
    "data": [
        "{FILE1}.xml",
        "{FILE2}.xml",
    ],
    "qweb": [
        "static/src/xml/{QWEBFILE1}.xml",
    ],
    "demo": [
        "demo/{DEMOFILE1}.xml",
    ],
```

```
    "post_load": None,
    "pre_init_hook": None,
    "post_init_hook": None,

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



See also:

- OCA's template: 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 *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.

On each version change a record in `doc/changelog.rst` should be added.

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

version in OCA

While OCA use odoo version in module version (e.g. `8.0.1.0.0`), we specify odoo version in *README.rst* file and use three numbers in version (e.g. `1.0.0`).

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 updatesfixes, 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 sytem application exists:

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

See also: *External dependencies in odoo*

doc/changelog.rst

Template

Use this for new modules

```
`1.0.0`
-------

- Init version
```

Guidlines

```
`2.0.0`
-------

- **NEW:** absolutely new way of ..

`1.2.0`
-------

- **NEW:** new interface for ..

`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

Notes

RST Requirements

Don't forget to keep correct rst format.

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.

Splited sentence 1.

Splited 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
```

Rendered RST

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

Splited sentence 1.

Splited 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
===========================

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

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"*.

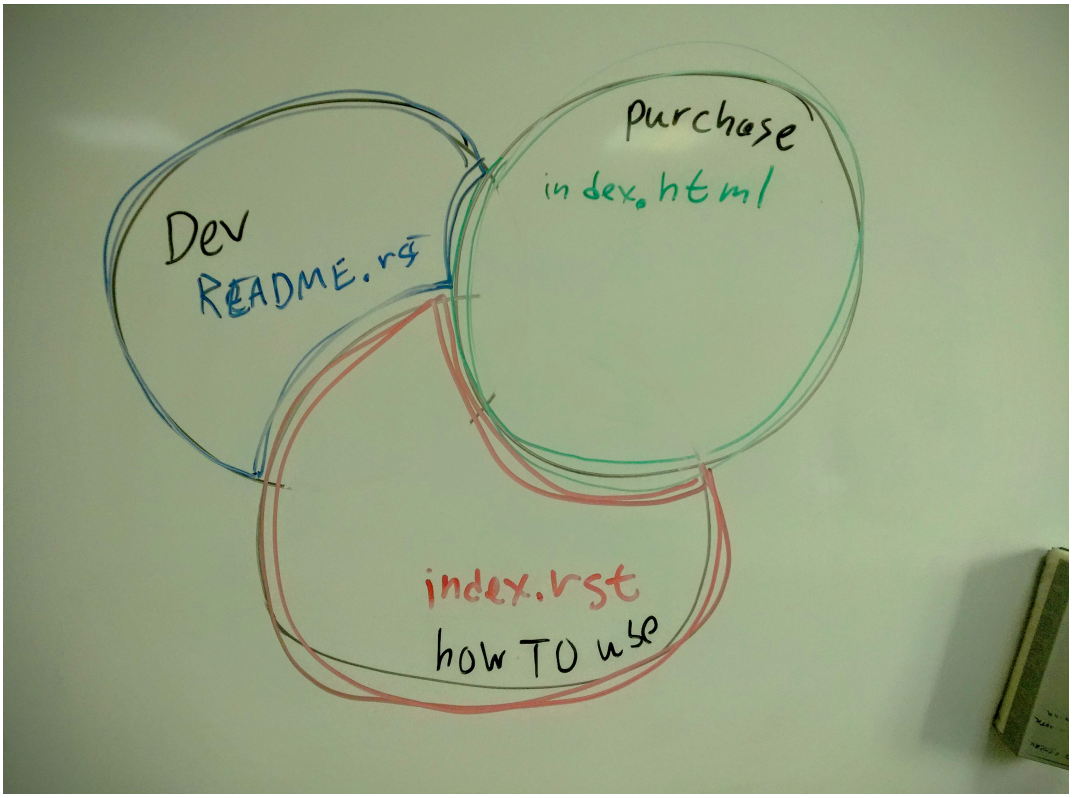
index.html

Module representation. It shall give an answer to the questions *"Do I need this module? Should I buy it?"*.

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



Template handling

Download templates:

```
cd PATH/TO/MODULE-ROOT/

# __manifest__.py
wget https://raw.githubusercontent.com/it-projects-llc/odoo-development/master/docs/
↪dev/docs/templates/__manifest__.py
# __README__.rst
wget https://raw.githubusercontent.com/it-projects-llc/odoo-development/master/docs/
↪dev/docs/templates/README.rst
mkdir doc
cd doc
# doc/index.rst
wget https://raw.githubusercontent.com/it-projects-llc/odoo-development/master/docs/
↪dev/docs/templates/doc/index.rst
# doc/changelog.rst
wget https://raw.githubusercontent.com/it-projects-llc/odoo-development/master/docs/
↪dev/docs/templates/doc/changelog.rst
```

```
cd ..
# empty __init__.py
touch __init__.py



# 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 "from . import main" >> controllers/__init__.py
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 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 REAMDE 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=odoo-telegram
REPO_NAME=pos-addons
REPO_NAME=website-addons

# Branch: choose one of the options
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-${ODOO_BRANCH}/odoo fetch upstream &&  export ODOO_REVISION=`git -
↪C ~/odoo/odoo-10.0/odoo rev-parse upstream/${ODOO_BRANCH}`


# Category: shoose 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="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 https://raw.githubusercontent.com/it-projects-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}/${DEVELOPER_NAME}/g" __manifest__.py README.rst doc/index.
↪rst
sed -i "s/{DEVELOPER_GITHUB_USERNAME}/${DEVELOPER_GITHUB_USERNAME}/g" __manifest__.py␣
↪README.rst doc/index.rst
sed -i "s/{REPO_NAME}/${REPO_NAME}/g" README.rst
sed -i "s/{BRANCH}/${ODOO_BRANCH}/g" README.rst
sed -i "s/{TECHNICAL_NAME}/${TECHNICAL_NAME}/g" README.rst
sed -i "s/{VERSION}/${ODOO_BRANCH}/g" README.rst
sed -i "s/{ODOO_COMMIT_SHA_TO_BE_UPDATED}/${ODOO_REVISION}/g" README.rst
```

```
#
```

Guidelines

Source:

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

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.

Odoo Python

Python decoratos

Original article

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

@api.one

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

something. For example:

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

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.

Pure Python

Compare two arrays

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

res.config.settings

Based on 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')
    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 ids, 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
```

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="[
        (4, ref('my.group'))
    ]"/>
</record>
```

module_XXX

Add XXX to the "depends" parameter in the `__openerp__.py` file

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)
```

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

Fields

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

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'
```

DateTime

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
- columns2: relational table right column name

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
```

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.

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,
                att_answer.id
        )""")
```

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 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 intenteded 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 repostitory to *addons-path*.

XML

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 modifyed. Record creating in module it declareted. 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>
```

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.

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()).

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.

HTML

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/>.

CSS

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;
}
```

YAML

Pure YAML

TODO

YAML in odoo

TODO

Javascript

Inheritance

TODO

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);
```

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;
```

```
        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, {}]);
    });
```

Frontend

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.

Point of Sale (POS)

Add new field in the model of POS module

To add new field in POS modules necessary in models.js override PosModel in the parent models which we take from "point_of_sale.models". For example:

```
var models = require('point_of_sale.models');
var _super_posmodel = models.PosModel.prototype;

models.PosModel = models.PosModel.extend({
    initialize: function (session, attributes) {
        // New code
        var partner_model = _.find(this.models, function(model){
            return model.model === 'product.product';
        });
        partner_model.fields.push('qty_available');

        // Inheritance
        return _super_posmodel.initialize.call(this, session, attributes);
    },
});
```

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,
```

Access

Security tutorial

Resources:

- 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

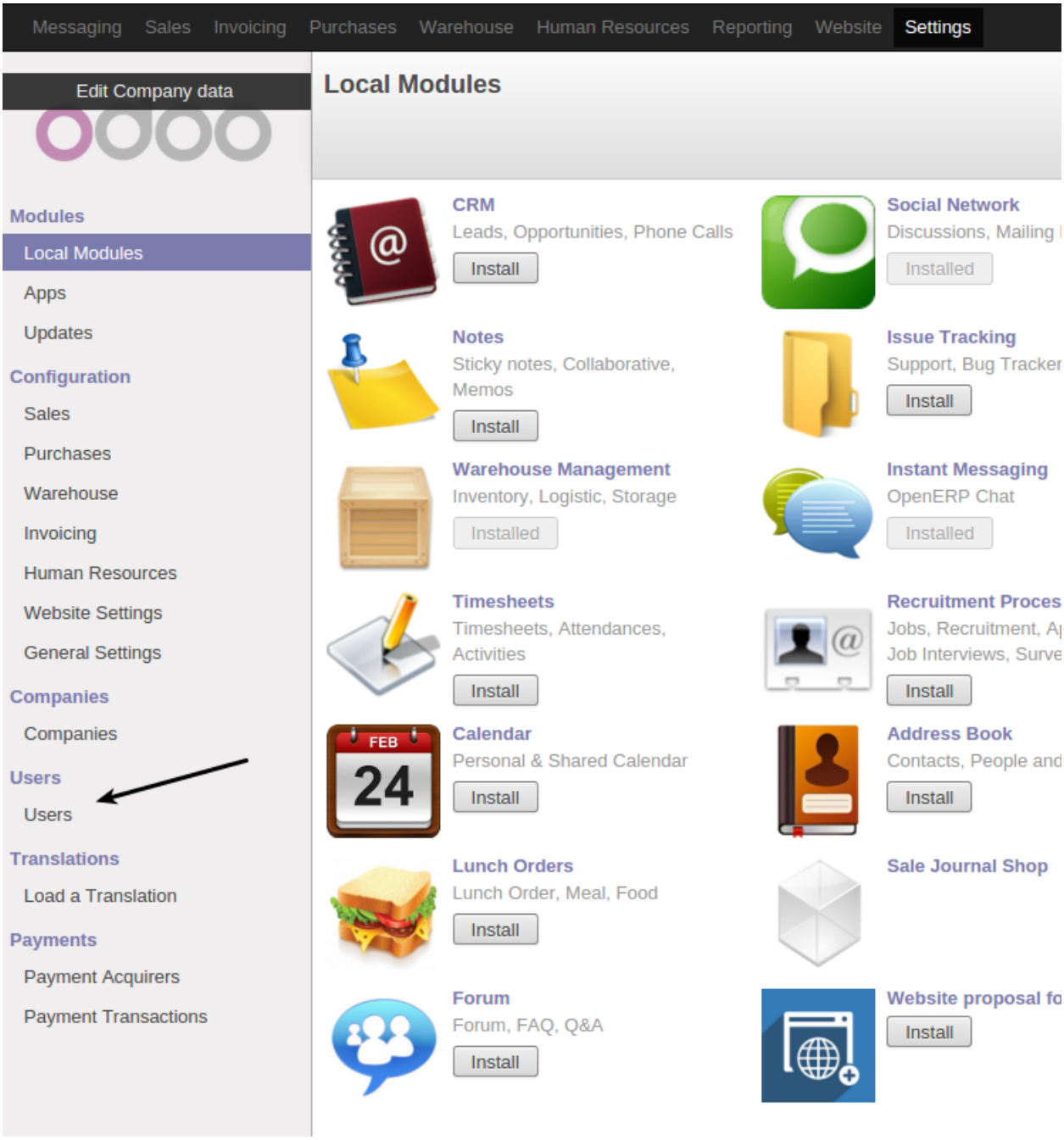
Odoo is very flexible on the subject of security. We can control what users can do and what they cannot on different levels. Also we can control independently each of the four basic operations: read, write, create, unlink. I.e. allow only read, allow only create, grant permission to create or delete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

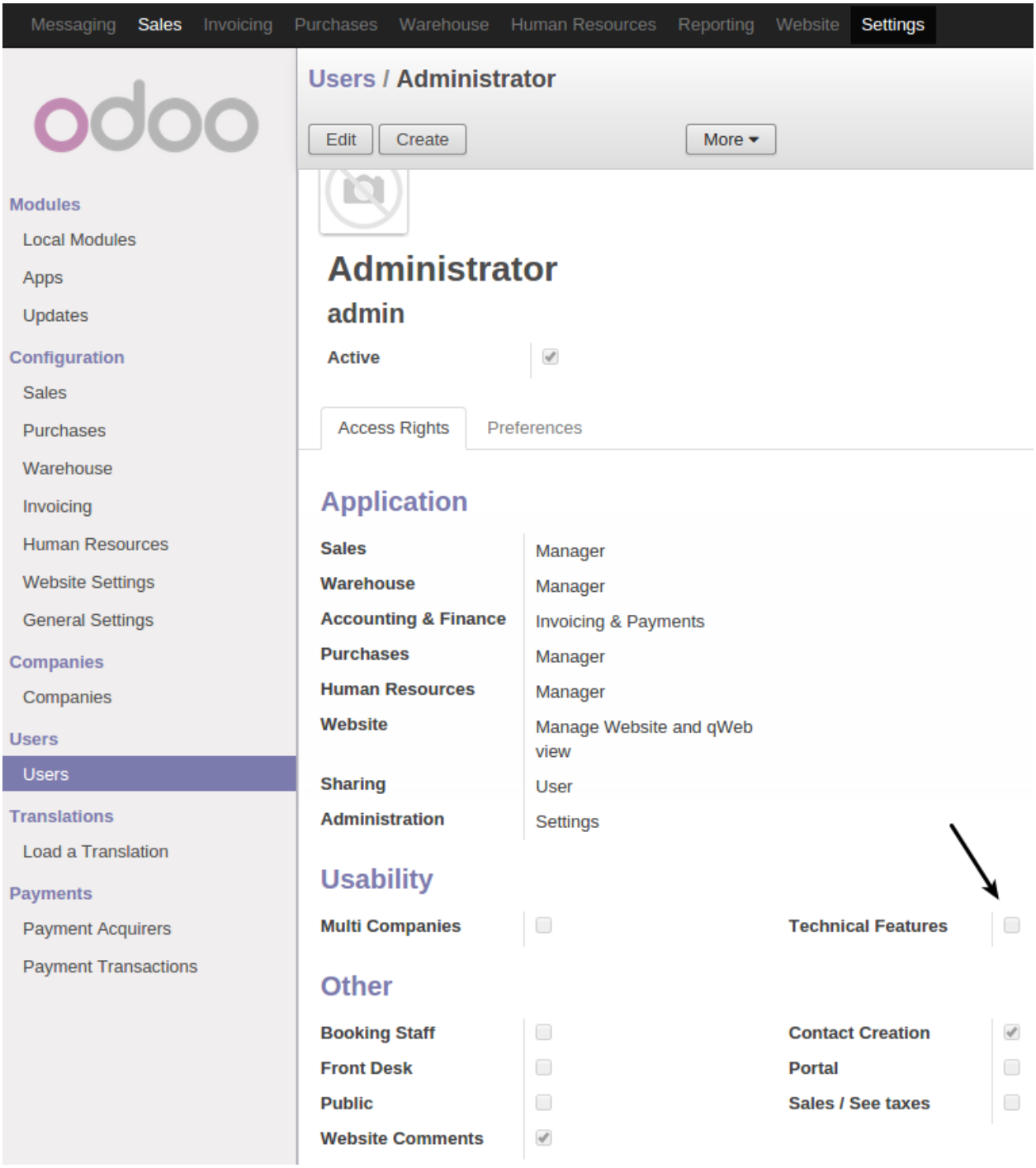
```
<record id="action_res_groups" model="ir.actions.act_window">
   <field name="name">Groups</field>
   <field name="type">ir.actions.act_window</field>
   <field name="res_model">res.groups</field>
   <field name="view_type">form</field>
   <field name="help">A group is a set of functional areas that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

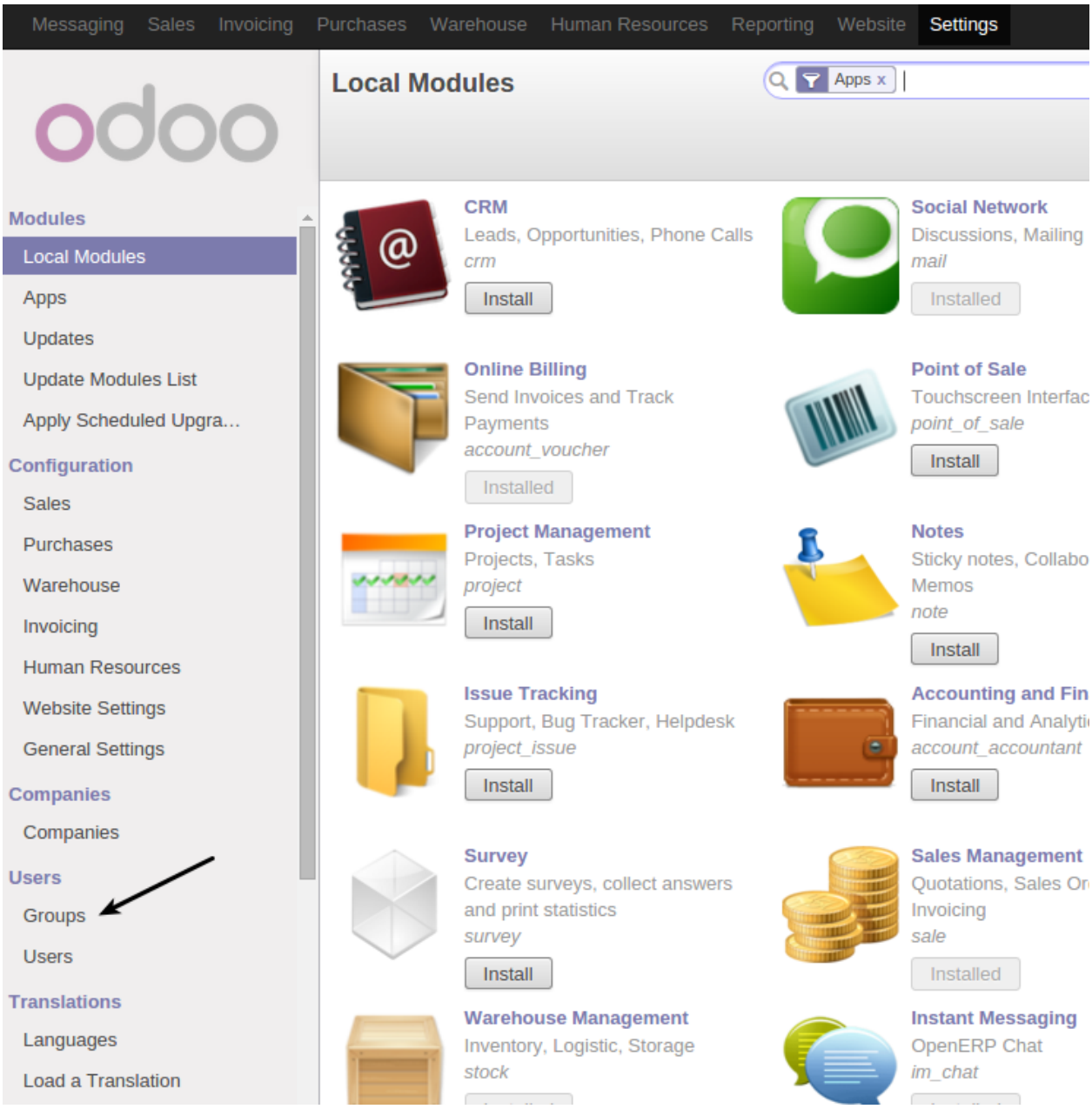
The `groups` attribute in the `menuitem` element shows us that only the members of `base.group_no_one` group can see the `Groups` menu item. The `base.group_no_one` xmlid is defined in the `./openerp/addons/base/security/base_security.xml` as follows.

```
<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.



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



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

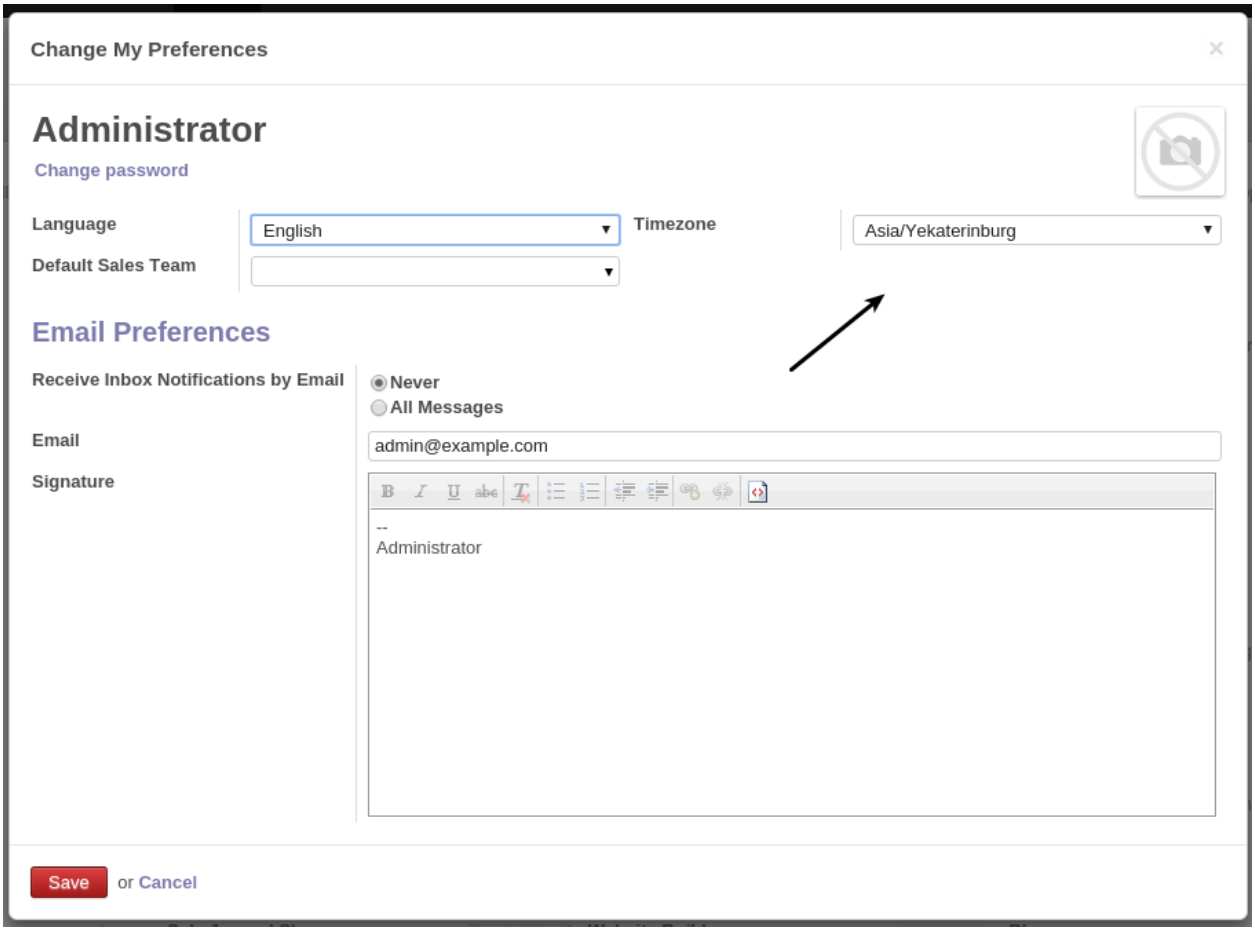
Hide fields

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>
```

```
    <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␣
↪edi ting 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

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.

Note: Official documentation states "record rules do not apply to the Administrator user although access rules do" seems to be wrong. Access Rules don't to apply to Administrator too. See the source: 8.0, 9.0, 10.0

See also:

- *ir.model.access*

- *ir.rule*

Hooks

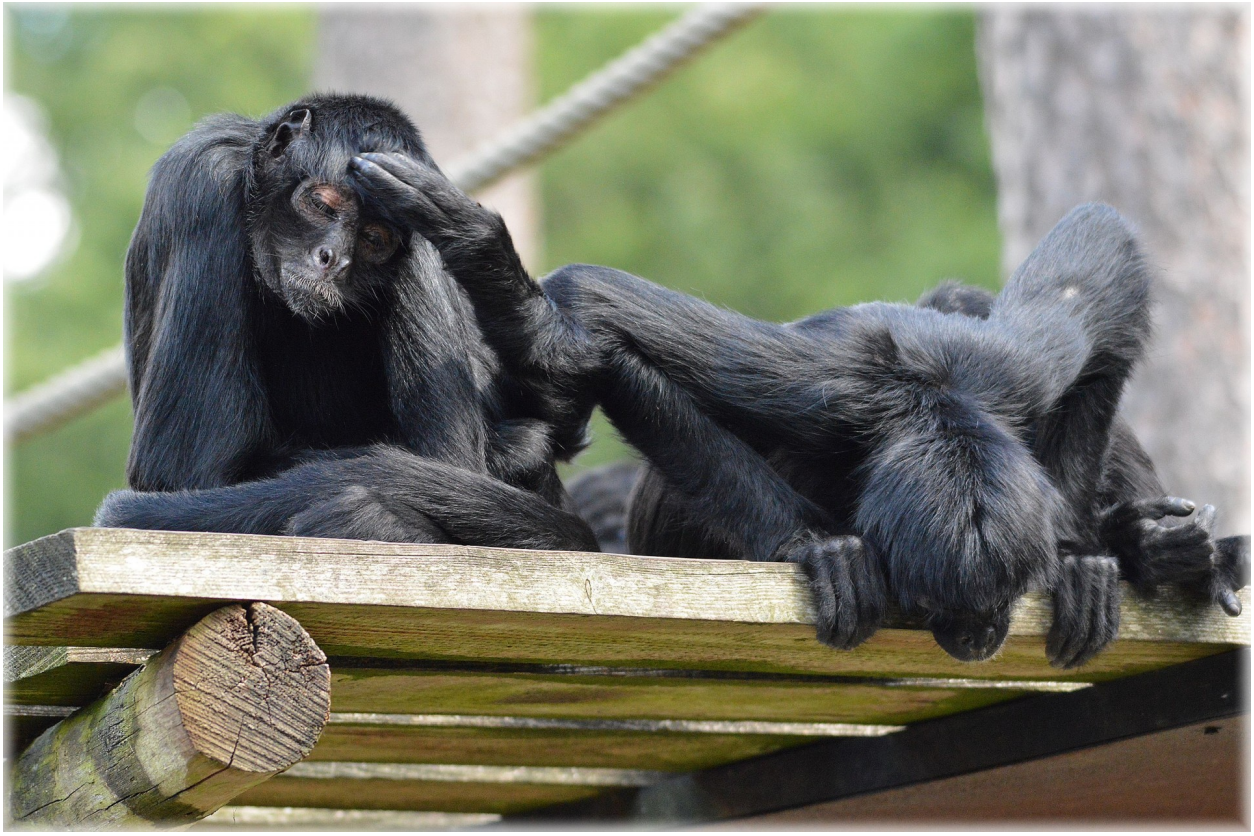
post_load

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(...)
    ...

tools.image_resize_images = new_image_resize_images
```

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

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
```

Tests

Basic python tests

How to run tests

This tests runs with `-d $DB_CONTAINER -u $MODULE --test-enable --workers=0` parameters.

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 a db which contains required modules (if you haven't got such db run new container with the key `-i` instead of `-u`. `-i` installs required module with its dependencies, whereas `-u` update already installed module)
- 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 \
-- -d $DATABASE_NAME -u $MODULE --test-enable --workers=0 --stop-after-init
```

How to make tests

To make some tests do next steps:

- Create folder named **tests**
- Add __init__.py file
- Create file that name begins from **test_**
- Add test methods that names start from **test_**

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

Example (will result testing error):

```
from odoo.tests.common import TransactionCase
class TestMessage(TransactionCase):
    at_install = False
    post_install = True
    def test_count(self):
        self.assertEqual(1, 0)
```

Test class

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.
    """
```

at_install, post_install

By default, odoo runs test with paramaters:

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

`at_install` - run tests right after loading module's files. It runs only in demo mode.

`post_install` - run test after full installation process. It differs from `at_install`, because

- it runs after calling `registry.setup_models(cr)`
- it runs after calling `model._register_hook(cr)`

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

JS Testing

Regular phantom JS tests

For automatic web tests odoo uses phantomjs.

How to write automatic js tests:

- Follow instruction for python tests
- In test method make call `self.phantom_js`

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

It is possible to run js phantom tests using *odoo tours* as JS testing code.

How to run tour in unittests:

- *Create tour* via js file
- Follow instruction for python tests
- run tour via phantom js
  - 10.0+:

```
self.phantom_js(
    URL_PATH,

    "odoo.__DEBUG__.services['web_tour.tour']"
    ".run('TOUR_NAME')",

    "odoo.__DEBUG__.services['web_tour.tour']"
    ".tours.TOUR_NAME.ready",

    login=LOGIN_OR_NONE
)
```

  - 8.0, 9.0:

```
self.phantom_js(
    URL_PATH,

    "odoo.__DEBUG__.services['web.Tour']"
    ".run('TOUR_NAME', 'test')",

    "odoo.__DEBUG__.services['web.Tour']"
    ".tours.TOUR_NAME",

    login=LOGIN_OR_NONE
)
```

How to run js tests

Additionally to general requirements, to run odoo with phantomjs tests:

- Install phantomjs or use dockers.
- use `--db-filter=.*`

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.

What to test

Obviously, you have to test features that module provide. But, it's important to have a stable module to test that features in a different context. This article tries to describe what that context could be. It can be used both for manual and automatic tests.

More about automatic tests:

- *Client-side unittests*
- *Server-side unittests*

User

While you develop a module, you can use an admin user for manual checking the result. It could simplify the process of development, because you can skip security stuff for a while. But when you prepare module for relase you absolutely need to check how system works from non-admin user.

> **Warning:** Admin user has *special access rights*. Use another User to test module.

Debugging

Logs

There are several places where you can get logs.

It's better to activate developer (debug) mode in browser when you are looging for logs.

- *Error Message*
- *Terminal*
- *Console*
  - *boot.js*
- *Sources*
- *Network*
  - *How to see html request initiator*

Error Message

It's a first place where you can see error message. But in most time, it doesn't contain enough information to resolve problem. Check other possbile ways to get log messages below.

Terminal

It's a place where you run odoo.

Any errors related to python can be found here

Console

It's a short term for browser's console. Click F12 in browser to open console.

It can contain error and warning about client part.

boot.js

Example is here: *Failed modules*

Sources

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

1. Turn on debug mode in the url.
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.

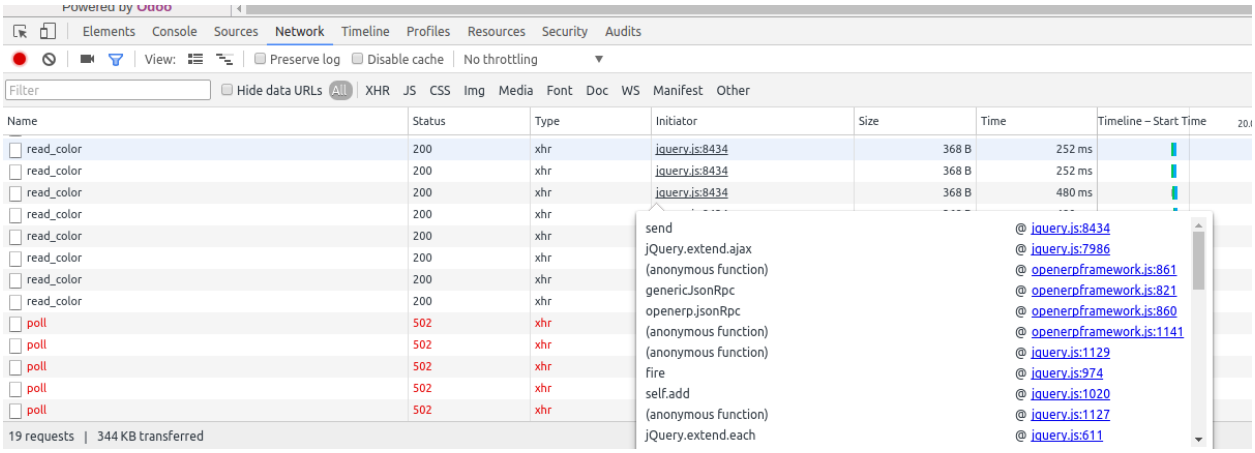
Example: *Missing dependencies error in console*

Network

Sometime error are not printed neither in Terminal, nor in Console. Then you can try to find some logs at Network tab of browser's developer tool. To see original odoo js files i.e. not minimized versions, *swich odoo in debug mode* first.

How to see html request initiator

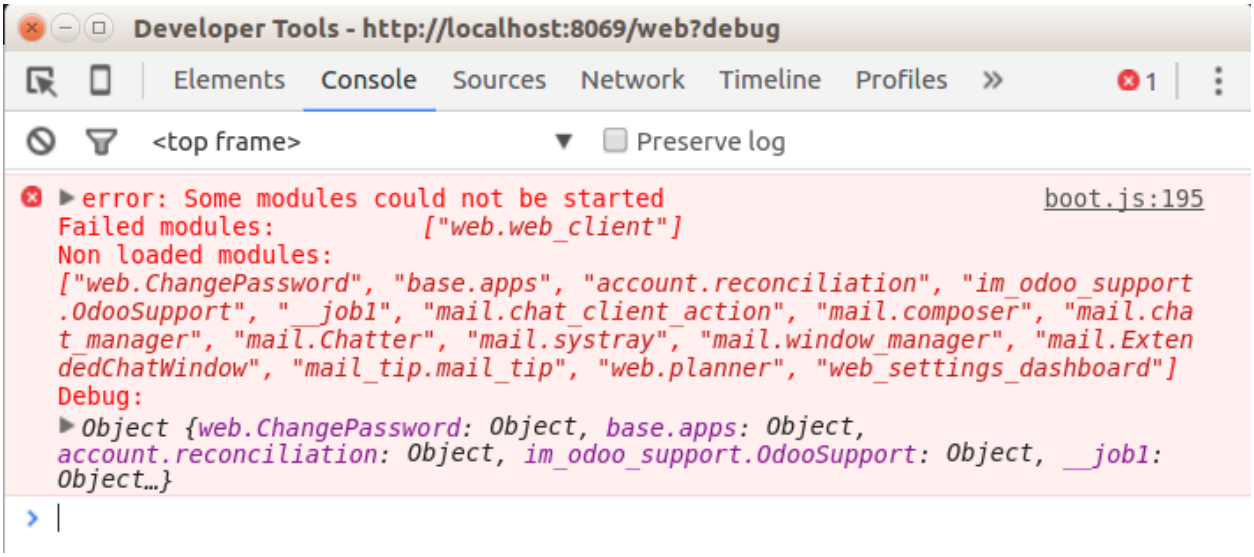
Suppose we want to know which part of our script initiate the request. If it is javascript we could see full program stack by putting mouse pointer on the initiator column's element.



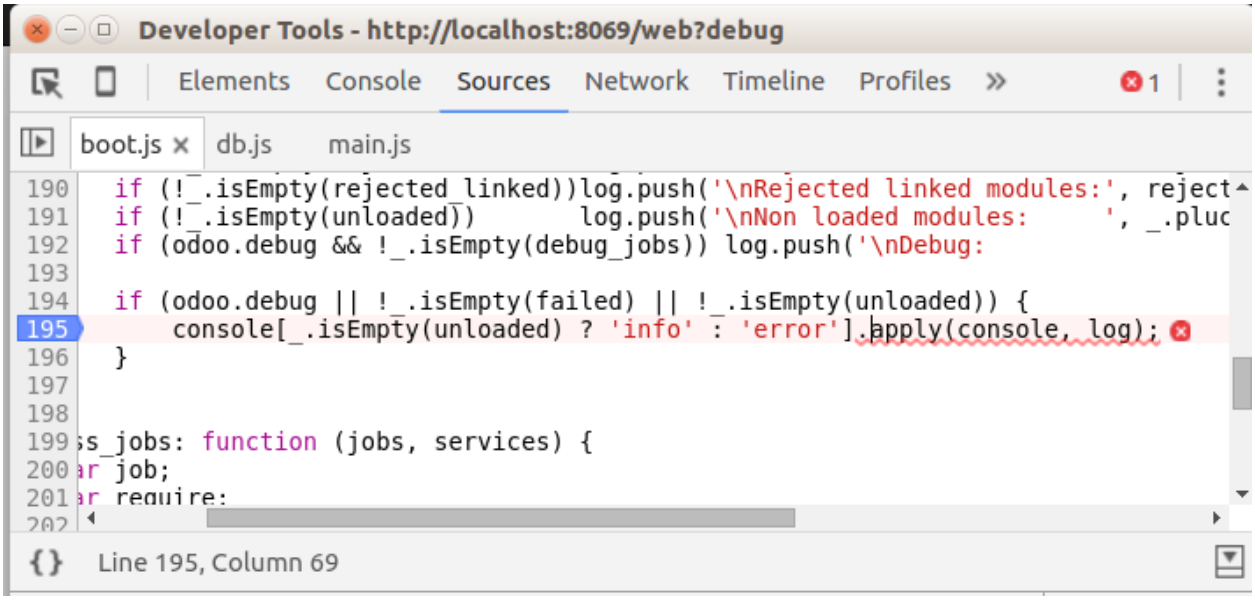
Typical errors

Error: Failed modules

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



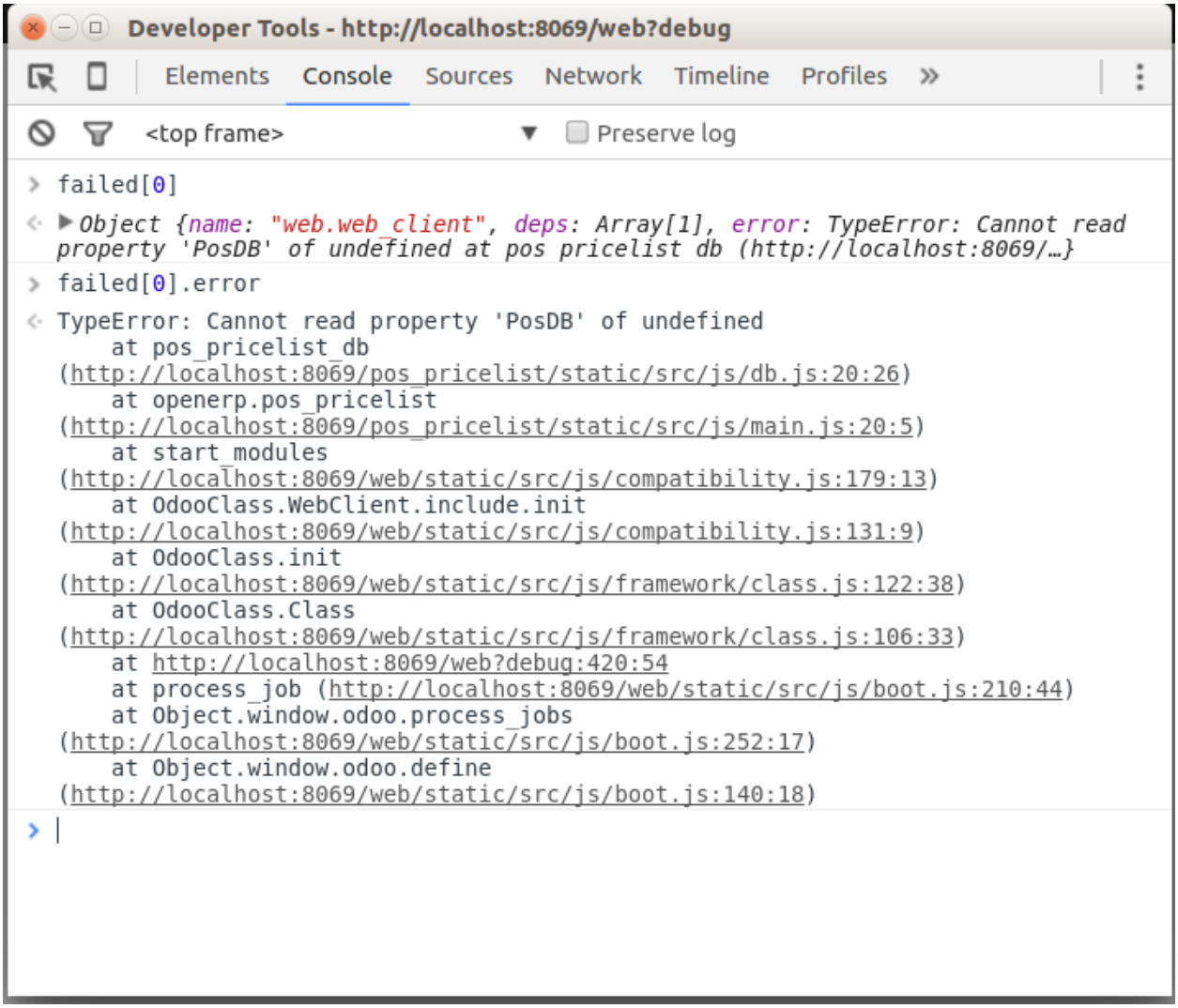
1. Go to error line into boot.js.
2. Turn on breakpoint.



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

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

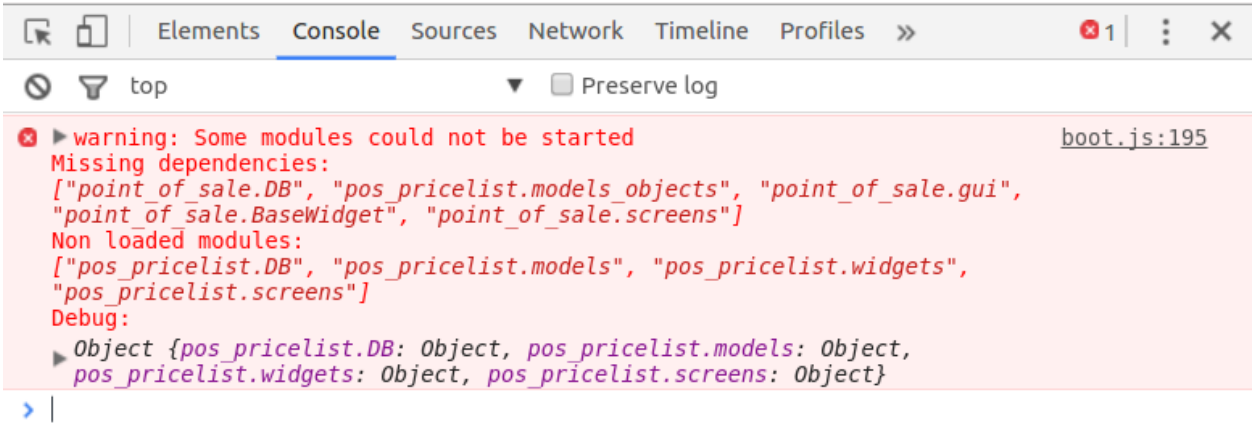
6. To receive the output



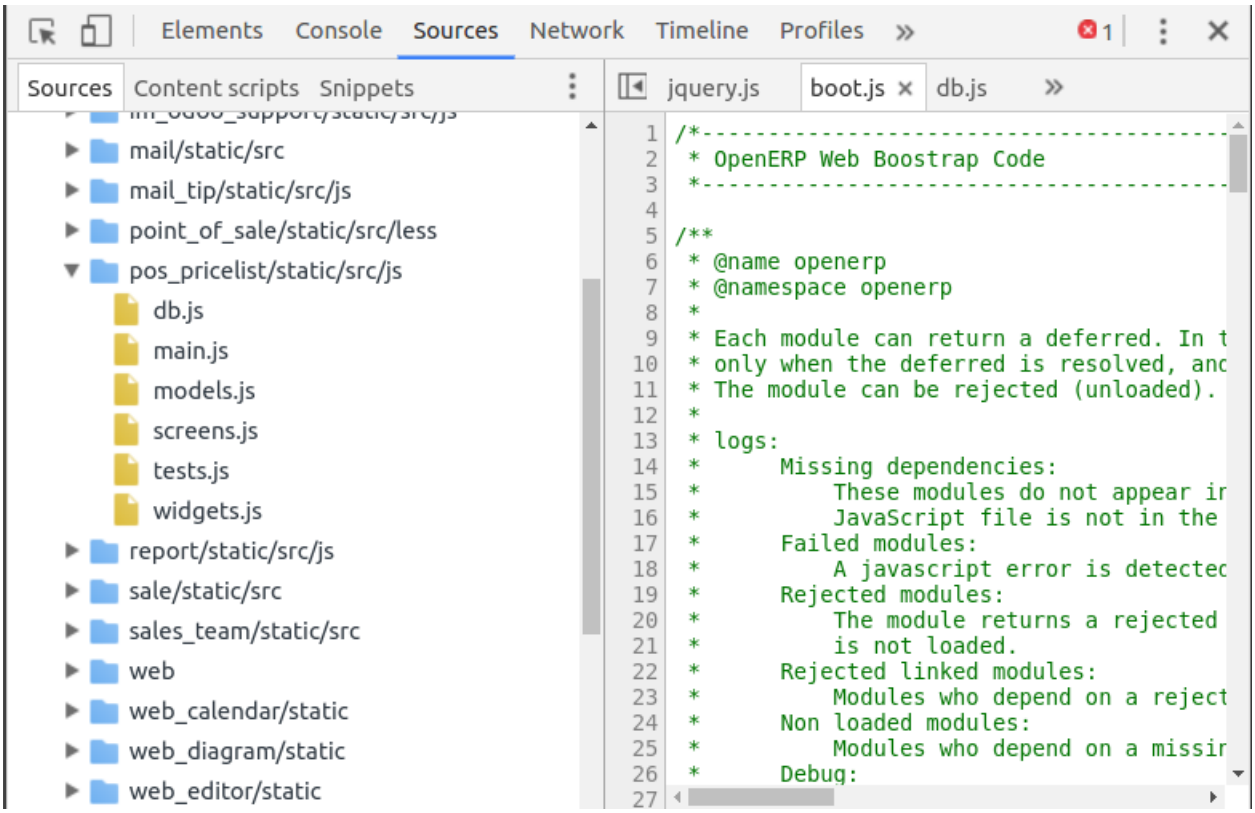
Error: Missing dependencies

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.

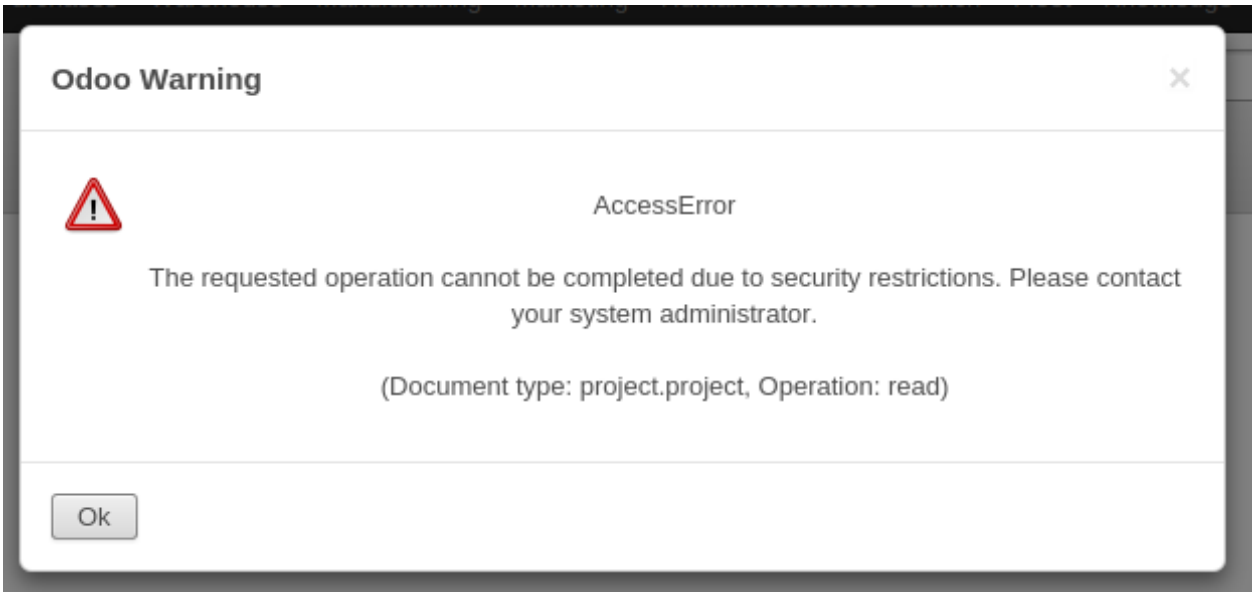


Likely you can not find files included in the Missing dependencies list. Then you need to check they are included in the view (.xml) files.

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.

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>
```

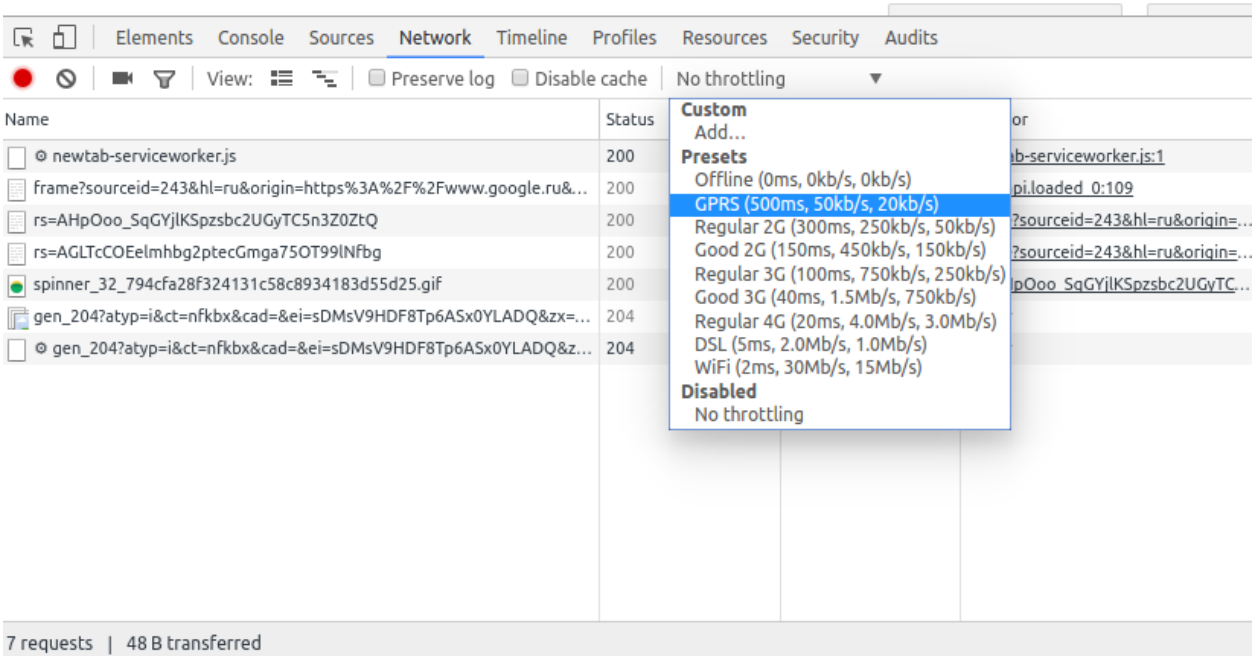
will stop execution if debugging is active (exact condition depend on the browser and its development tools)

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

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"
# Probabilyt 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
```

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 '3333333333338'| grep -o . | xargs xdotool key && xargs xdotool key␣
↪\n &
```

Where: 3 - sleep seconds; 3333333333338 - barcode.

After successfully scanning you will see '3333333333338' 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.

ESC/POS printer emulation

hw_escpos

- apply patch

```
cd /path/to/odoo/

# odoo 10
curl https://raw.githubusercontent.com/it-projects-llc/odoo-development/master/
↪docs/dev/debug/hw_escpos-patch/hw_escpos-10.patch > hw_escpos.patch

# odoo 9
curl https://raw.githubusercontent.com/it-projects-llc/odoo-development/master/
↪docs/dev/debug/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

Source Diving

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

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>`
```

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

Lint

Script for fixing travis error on odoo

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
```

Script

```
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'

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

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

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


# 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'

# 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'

#lint for js:
```

```
fixmyjs --legacy --config ~/.jshintrc ./

# Addition of the first row (coding) in py-files
find -iname '*.py' | xargs grep -rLP 'coding: *utf-8' | xargs sed -i '1s/^/# -*-␣
↪coding: utf-8 -*-\n/'

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

# Duplicate implicit target name: "changelog".
find . -type f -name 'changelog.rst' | xargs sed -i 's/^Changelog/Updates/g'
find . -type f -name 'changelog.rst' | xargs sed -i 's/^=========/=======/g'

# Replace @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/'\
'@api\.one\n'\
'    def ([^(]*)\(self, ([^(]*)\):/'\
'@api.multi\n'\
'    def $1(self, $2):\n'\
'        for r in self:\n'\
'            r.$1_one($2)\n'\
'        return True'\
'\n'\
'\n'\
'    \@api.multi\n'\
'    def $1_one(self, $2):\n'\
'        self.ensure_one()/g'

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

Run following script only once:

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

Other

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

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

    def _create_view(self, cr, uid, name, inherit_id, arch, noupdate=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': noupdate,
        })
        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])
```

Odoo database

Many to many

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

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 adons 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

Module Migration

Switching module to new api

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.except_osv/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(\(.*\) \(["\x27][^,]*\)/
↪fields.function(\1 string=\2/g'
find . -type f -name '*.py' | xargs sed -i 's/fields.function(\(.*\) multi=[^,)]*/
↪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'
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.\u\1/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'
```

Semi-Automatic replacements

We recommend to use commands below after commiting 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'
```

10.0+ updates

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

Fixing references on migration

9.0− → 10.0+

```
# 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'
```

Migration to python3

```
# TODO
```

Quick source review

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
```

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
- https://www.it-projects.info/page/module-migration

CHAPTER 4

User documentation

Module releasing checklist

This articles cover documentation and description part only.

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*

Summary

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

Price

- Review `"price"` attribute at *manifest file*

Category

- Review `"category"` attribute at *manifest file*

doc/index.rst

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

README.rst

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

static/description/index.html

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

Main image

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

static/description/index.html

Image sizes

- *Image Sizes*

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%;
↪">
      <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">SUBSECTION NAME</h4>
```

(Put it inside <section class="..."> tags)

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">
                <img class="oe_demo oe_picture oe_screenshot" src="IMAGE.png"/>
            </div>
        </div>
    </div>
</section>
```

Image + Text

TODO

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">
             <img class="oe_demo oe_picture oe_screenshot" src="IMAGE.png"/>
         </div>
    </div>
</section>
```

Free Support

```
<section class="oe_container">
  <div class="oe_row oe_spaced">
    <h2 class="oe_slogan" style="color:#875A7B;">Free Support</h2>
    <h3 class="oe_slogan">You will get free support in case of any issues</h3>
  </div>
</section>
```

Contact us

- *Contact us block*

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>
```

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
```

Module description

Main screenshot

The main screenshot displayed only in Odoo Apps 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/main-screenshot.png']
```

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.

Summary

This is an overview of content that provides a reader with the overaching 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"""
```

Contact us block

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

```
export ODOO_BRANCH=10.0
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
```

JS Tour

Tours are used to demonstrate module capabilities step by step with popup windows. It may be launched automatically or manually.

Creating Tour

10.0+

Example from website_sale module:

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

var tour = require("web_tour.tour");
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',
        },
        /* ... */
    ]
);

});
```

Each step may have following attrubutes:

- **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*
- **position** – how to show tip (left, rigth, top, bottom), default right
- **width** – width in px of the tip when opened, default 270
- **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

Options (second argument of `tour.register`):

- **test** – only for tests
- **url** – open link before running the tour
- **wait_for** – wait for deffered object before running the script
- **skip_enabled** – adds *Skip* button in tips

More documentation:

- https://www.odoo.com/slides/slide/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/tip.js

8.0, 9.0

Tour is a simple JS file with some determined structure. 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") },
        },
        {
            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',
        },
    ]
}
```

What you do here is describing steps that got to be proceeded by user or phantom (phantomjs).

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

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>
```

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

You can launch tour by entering in browser address like this mydatabase/web#/tutorial.mails_count_tour=true where after tutorial. is id of your tour.

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
```

Launch Tour 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&amp;model=res.partner&amp;/#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&amp;res_partner_mails_count=tutorial#id=
↪'+str(ref('base.partner_root'))+'&amp;view_type=form&amp;model=res.partner&amp;/
↪#tutorial_extra.mails_count_tour=true'"/>
```

Preview module on App Store

Browser's dev tools allows to preview Module in App Store before actual uploading.

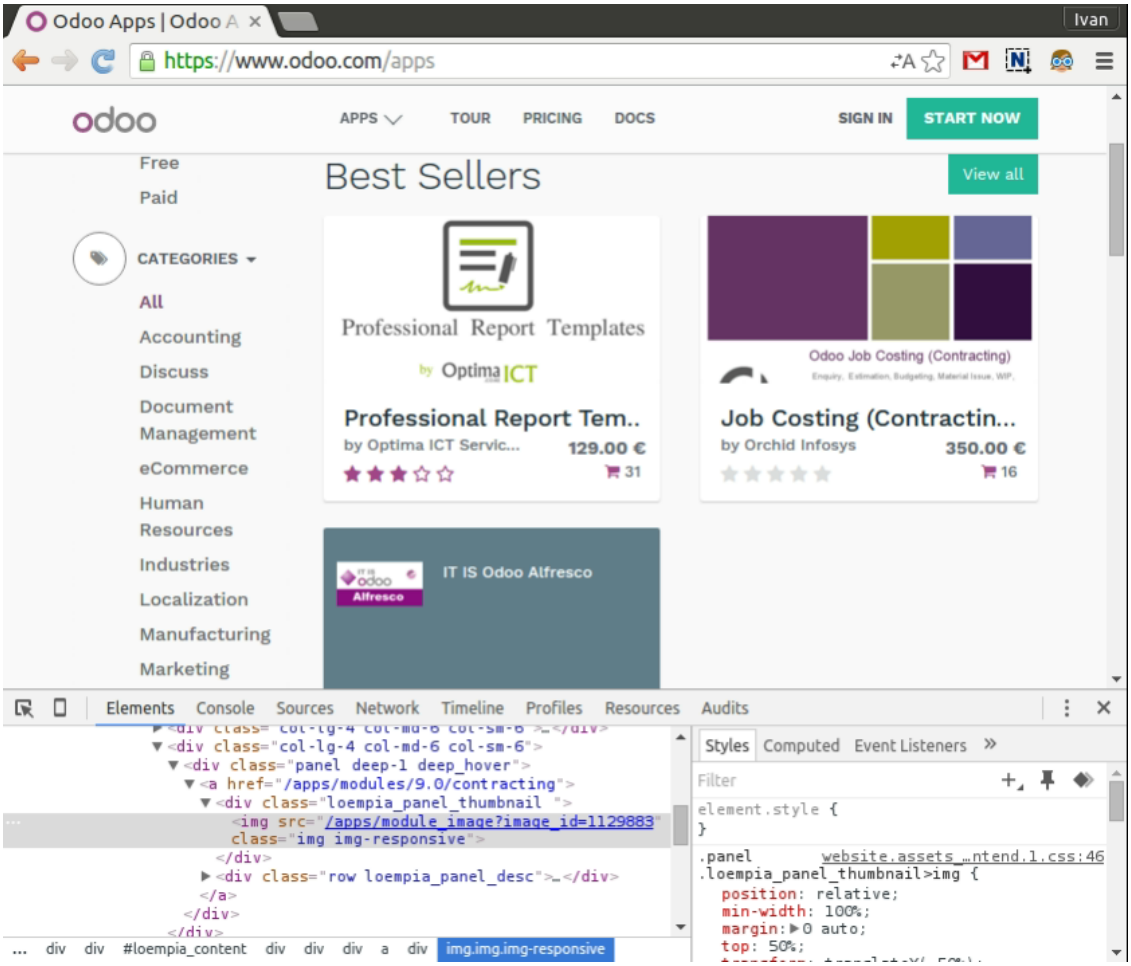
- open https://www.odoo.com/apps
- click Inspect Element on some application
- change text and images
- Done! Now can decide do you need make changes or keep current images and text

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 `......AAElFTkSuQmCC
```

- paste this line to src attribute of image tag
  - BEFORE
```

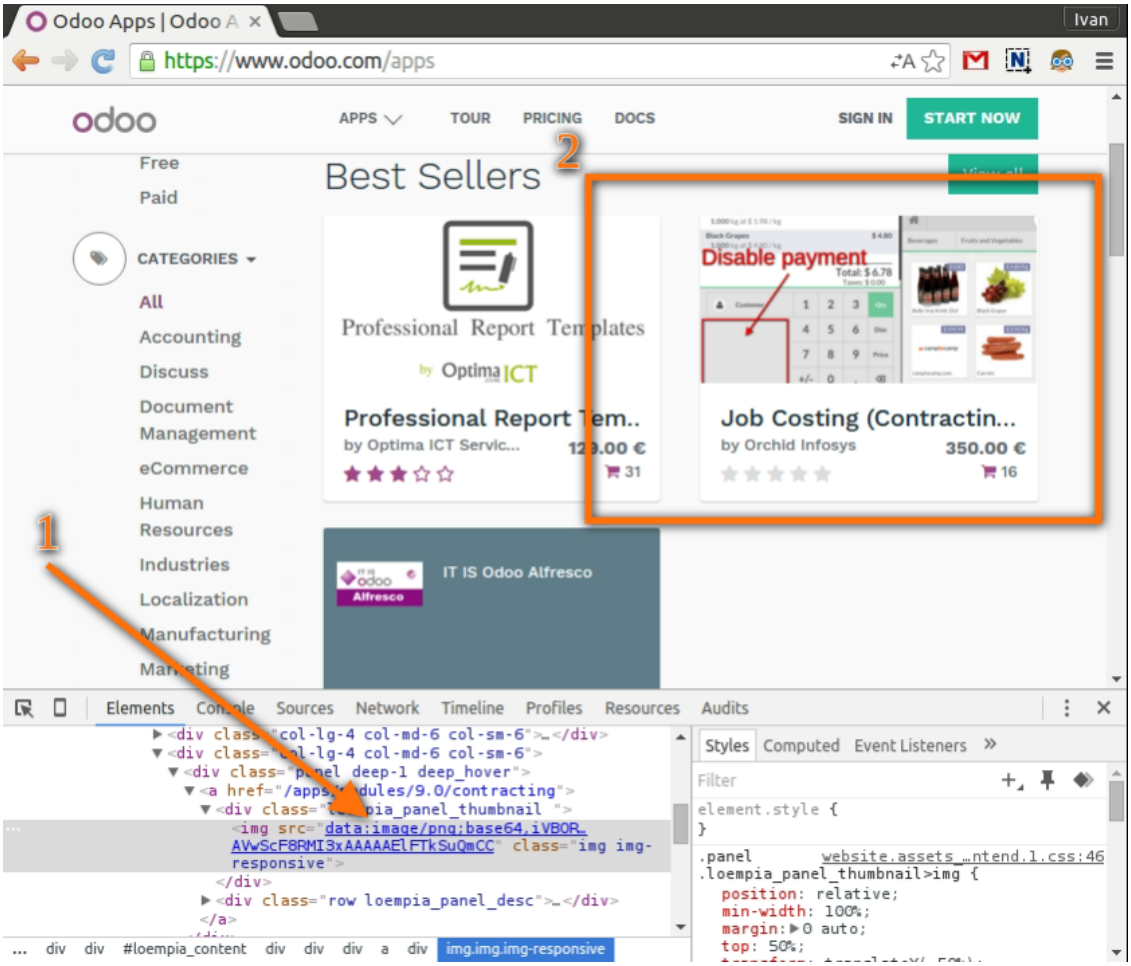

Odoo Apps | Odoo A
Ivan
https://www.odoo.com/apps
odoo
APPS
TOUR
PRICING
DOCS
SIGN IN
START NOW
Free
Paid
CATEGORIES
All
Accounting
Discuss
Document Management
eCommerce
Human Resources
Industries
Localization
Manufacturing
Marketing
Best Sellers
View all
Professional Report Templates
by Optima ICT
Professional Report Tem..
by Optima ICT Servic...
129.00 €
31
Odoo Job Costing (Contracting)
Job Costing (Contractin...
by Orchid Infosys
350.00 €
16
IT IS Odoo Alfresco
Elements
Console
Sources
Network
Timeline
Profiles
Resources
Audits

<img src="/apps/module_image?image_id=1129883"
class="img img-responsive">

…

Styles
Computed
Event Listeners
Filter
element.style {
}
.panel website.assets_…ntend.1.css:46
.loempia_panel_thumbnail>img {
position: relative;
min-width: 100%;
margin: 0 auto;
top: 50%;
... div div #loempia_content div div div a div img.img.img-responsive

– AFTER



Nginx

Configure your nginx and use local link in src attribute.:

```
<img src="static.local/path/to/image.png"/>
```

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

Image sizes

See also

- *Preview module on App Store*
- *Adjust chromium window size script*

__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.

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">
            <img class="img img-responsive" src="1.png"/>
        </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;

CHAPTER 5

Git and Github

Initial git & github configuration

ssh keys

Configure github ssh keys: https://help.github.com/articles/connecting-to-github-with-ssh/

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
```

```
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 be "Unverified"

github profile

IT-Projects LLC employees only:

- https://github.com/settings/profile
  - public email must be personal address ... @it-projects.info
  - URL must be link to public profile at company's website
  - Company must be set to `@it-projects-llc`
  - Location must be your `YouCity, Country`
  - Photo must be your real face photo
- https://github.com/settings/emails
  - primary email must be personal address ... @it-projects.info
  - "Keep my email address private" must be switched off
- https://github.com/orgs/it-projects-llc/people
  - get invitation
  - set `Organization visibility` to `Public`

git email

- Configure email in git. Email must be the same as in github settings:

```
git config --global user.email "your_email@example.com"
```

git editor

```
git config --global core.editor "nano"
```

gitignore

- Configure global gitignore

  Possible content for `~/.gitignore_global`:

```
*~
*.pyc
```

Porting

If you add some feature to one branch and need to add it to anoher branch, then you have to make *port*.

See also:

- *Conflicts resolving*

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
```

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:

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 betwwen 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/it-projects-llc/odoo-saas-tools/pull/286/commits , use command:

```
git cherry-pick 6ee4fa07d4c0adc837d7061e09da14638d8abf8d^..
↪9133939a25f9e163f52e6662045fc2dc6010ac14
```

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 use. 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
  ```

Deleted files

Sometimes, changes can be conflicted because files are not exist anymore in *ours* version, but updated in *theirs*. 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
```

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 mergingchery-picking commits be only about merging and chery-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.

Multi Pull Request

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-projects.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-projects.info>
# Date:   Thu Feb 25 11:31:43 2016 +0500
#
#     Merge remote-tracking branch 'origin/8.0' into 9.0-dev
#
# commit 6981c245afdccc39b2b49585f8205a784161f9c6
# Merge: 22081ed 6eb9f8d
# Author: Ivan Yelizariev <yelizariev@it-projects.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
```

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

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.

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
```

Files relocation

- *git format-patch*
- *git filter-branch*

git format-patch

This section is based on OCA's instruction.

Used variabes:

- `$REPO_PATH`, `$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 $REPO_PATH
git fetch upstream
git format-patch --stdout --root upstream/$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
```

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 perserve 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
       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 movemodule.sh file in your home directory and put all lines above there and make this file executable.

```
$ cd ~
$ chmod +x movemodule.sh
```

To do the moving of group_menu_no_access from addons-yelizariev to access-addons with the movemodule.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-yelizarie$ ~/movemodule.sh group_menu_no_access ~/access-addons 8.0
```

Commit comment prefix

Based on: https://www.odoo.com/documentation/8.0/reference/guidelines.html

Basic tags

- **[DOC]** for documentation. Don't use any other tags when you improve, fix, refactor documentation
- **[PORT]** for porting *(version tag is required)*
- **[BACKPORT]** for back-porting *(version tag is required)*
- **[IMP]** for improvements
- **[FIX]** for bug fixes
- **[REF]** for refactoring
- **[TEXT]** for commits with text changes only: labels, hints, comments, etc., but not for updates in documentation (*.rst and *.html files)
- **[NEW]** for uploading new modules *(version tag is required)*
- **[ADD]** for adding new resources and features.
- **[REM]** for removing of resources
- **[DEMO]** for adding / updating demostration data
- **[CI]** for updating `.travis.yml`, `requirements.txt`, `*/tests/*`, etc. files
- **[LINT]** for fixing lint errors
- **[i18n]** for translations

Version tags

- **[8.0]**
- **[9.0]**
- **[10.0]**
- etc.

Put tags before other tags, e.g.:

```
[9.0][BACKPORT] module_xxx
```

Temporar tags

- **[WIP]**, **[DEV]**, **[TMP]** – for commits that has to be squashed before merging

Which tag to use?

- If commit upload new module:
  - use **[NEW]**
- If commit updates are only in module description (*doc/**, *static/description/**, *README.rst*, `name` and `summarry` attributes at manifest) or in code comments:
  - use **[DOC]**
- If commit works with translationlocalisations only:
  - use **[i18n]**
- If commit changes only some string related to UI (e.g. Error Message, Name of something etc.)
  - use **[TEXT]**
- If commit fixes issue related to switch to another major odoo version
  - use *Version tag* and
    - use **[PORT]** if target version is newer than original (e.g. porting from odoo 10.0 to odoo 11.0)
    - use **[BACKPORT]** if target version is older than original (e.g. porting from odoo 10.0 to odoo 9.0)
- If commit updates demostration data
  - use **[DEMO]**
- If commit updates / configures automatic tests
  - use **[CI]**
- If commit fixes existing features:
  - use **[FIX]**
- If commit improves existing features:
  - use **[IMP]**
- If commit adds new features:
  - use **[ADD]**
- If commit makes updates asked by lint tools:
  - use **[LINT]**
- If commit updates (refactors) existing code without adding or fixing features:
  - use **[REF]**

Notes

- Order of this *if-then-that* list matters. Use some *that* only if all *if-blocks* above it are false.
- If your commit makes update of different types you need to describe each update separately using appropriate tag, e.g.:

```
[FIX] bug in feature X ...
[IMP] UI improvements in feature Y ...
...
```

Emoji

Yes, you can add some fun and emotions to your commit. Select emojii you like here and copy-paste to you commit message, e.g.:

```
[NEW] super-puper module is released! :boom:
```

Git stash

- book: https://git-scm.com/book/no-nb/v1/Git-Tools-Stashing
- man: https://git-scm.com/docs/git-stash

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
```

Squash commits into one

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
```

git commit --amend

Instead of creating new commit, adds updates to the latest commit.

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 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.

Push

```
git push -f origin 9.0-new-module
```

CHAPTER 6

Continuous Integration

Runbot

runbot.odoo.com

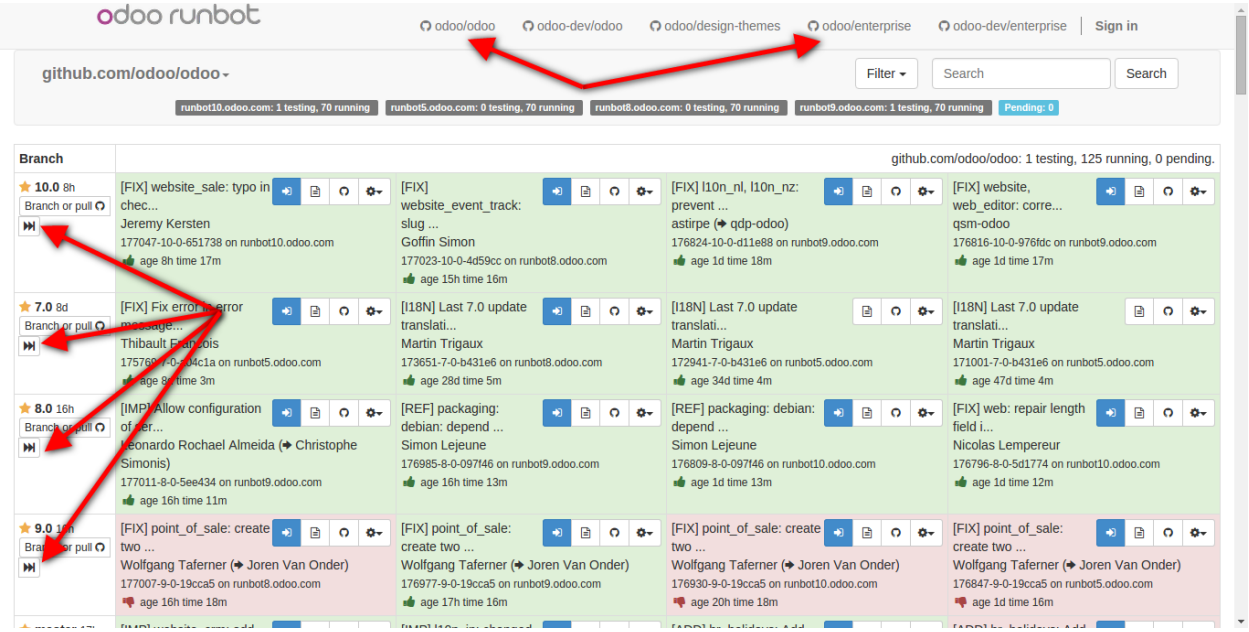
http://runbot.odoo.com/ – official runbot. While its main purpose is checking pull requests to official repository, it is usefull on daily development routine.

- 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`



runbot.it-projects.info

http://runbot.it-projects.info/ – customized runbot for IT-Projects' repositories.

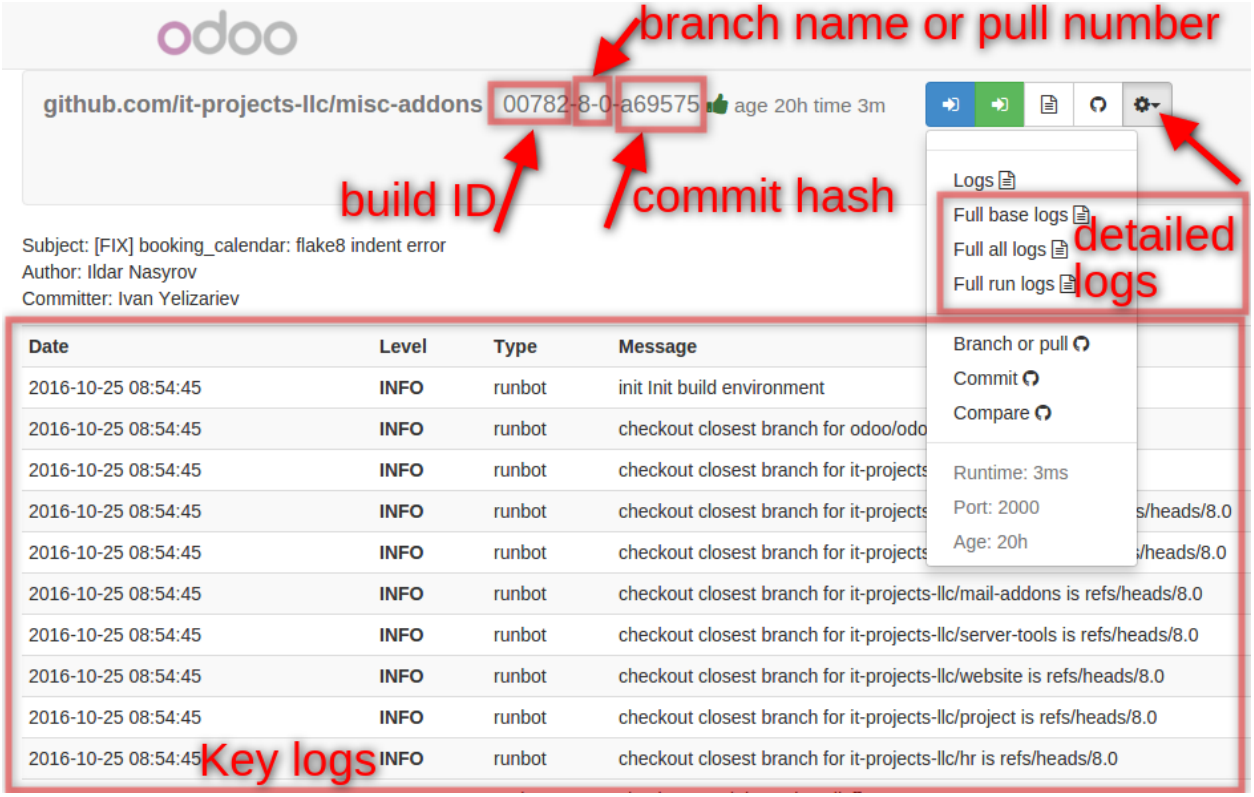
Stages of making a build:

- 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



How to deploy runbot?

There is docker that allows you deploy you own runbot for your repositores. Check it out for further information

- https://github.com/it-projects-llc/odoo-runbot-docker

Odoo Travis Tests

TODO

Coverage

CHAPTER 7

Odoo

Models

Section helps in understanding built-in models

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 manulally

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 manulally

Cons:

- record is not deleted on uninstalling
- value is overwrited 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 overwrited if it already exists

Cons:

- record is not deleted on uninstalling

res.users

TODO

res.groups

TODO

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`)

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')
```

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

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)
```

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).

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

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.

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

- 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

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="doal">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="doal">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

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.

How to use Odoo

How to create database

From UI

To create new database open `/web/database/manager`

8.0-

Database with dots

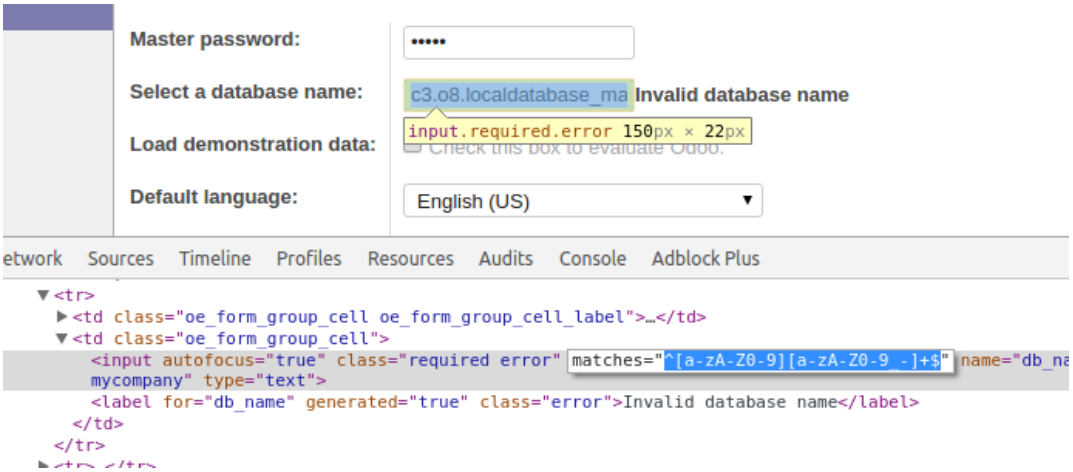
Early version of odoo doesn't allow to create databases with dots. You can remove this restriction in two ways:

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`

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*.

How to install/update module

There are several ways to installupdate module

From App store (install)

10.0+

- *activate developer mode*
- navigate to `Apps` menu
- click on `App Store` menu in left side bar
- remove `Featured [x]` filter
- search module you need
- click `[Install]`

9.0

- *activate developer mode*.
- navigate to `Apps` menu
- click on second `Apps` menu in left side bar
- remove `Featured [x]` filter
- search module you need
- click `[Install]`

8.0

- navigate to `Settings / Modules / Apps`
- remove `Featured [x]` filter
- search module you need
- click `[Install]`

From App store (update)

10.0+

- navigate to `Apps` menu
- click `Updates` in left side bar

9.0

- navigate to `Apps` menu
- click `Updates` in left side bar

8.0

- navigate to `Settings / Modules / Updates`

From zip archive (install)

- unzip module to your addons folder
- restart odoo server

10.0+

- *activate developer mode*
- navigate to `Apps` menu
- click `Update Apps List`
- click `Apps` in left side bar
- search and open a module you need
- click `[Install]`

9.0

- *activate developer mode*
- navigate to `Apps` menu
- click `Update Apps List`
- click `Apps` in left side bar
- search and open a module you need
- click `[Install]`

8.0

- navigate to `Settings / Modules`
- click `Update Modules List`
- click `Local Modules` in left side bar
- search and open a module you need
- click `[Install]`

From zip archive (update)

- unzip and replace module in your addons folder
- restart odoo server

8.0

- navigate to `Settings / Modules`
- click `Local Modules` in left side bar
- search and open a module you need
- click `Upgrade`

9.0

- *activate developer mode*
- navigate to `Apps` menu
- search and open a module you need
- click `Upgrade`

10.0+

- *activate developer mode*
- navigate to `Apps` menu
- search and open a module you need
- click `Upgrade`

How to activate developer mode

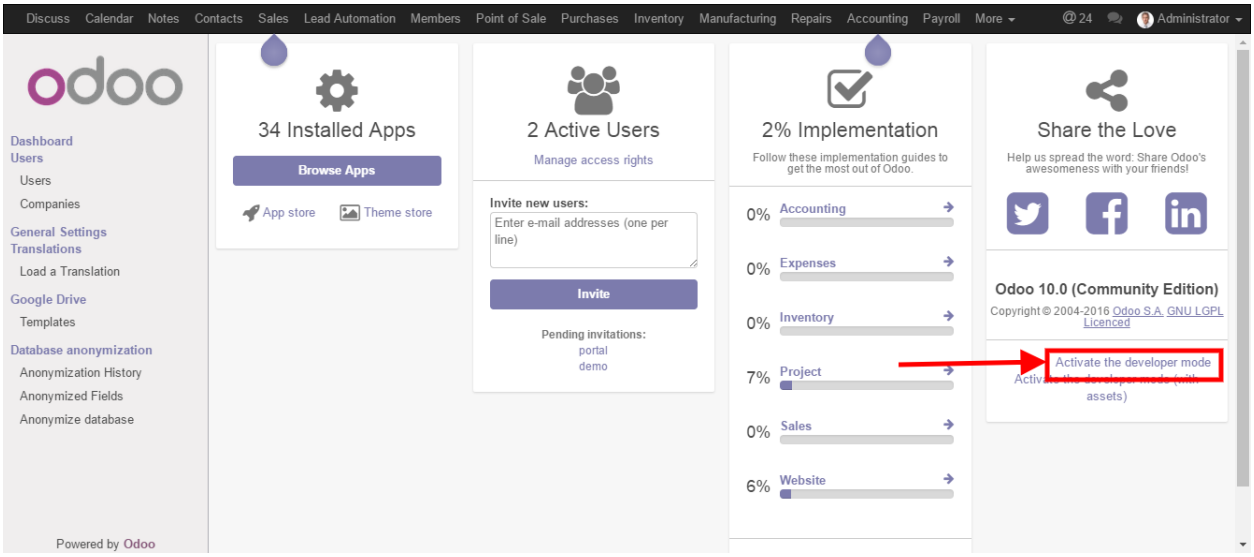
1. Add `debug` parameter to your url, e.g.:

```
localhost:8069/web?debug=1
```

2. Or use UI

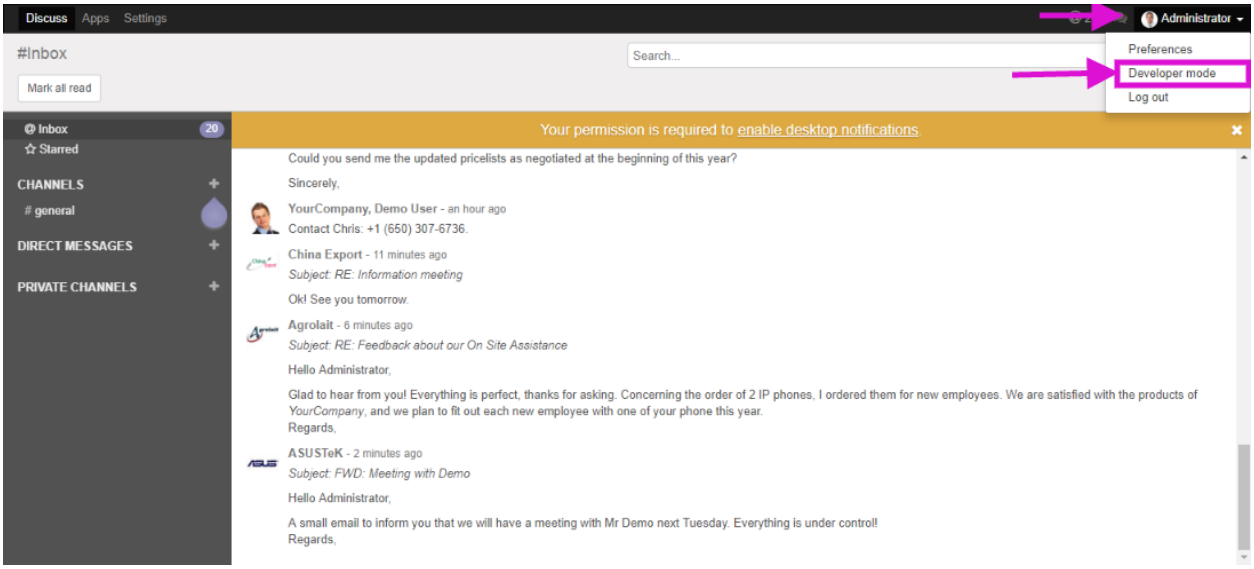
10.0+

- go to `Settings`
- click `Activate the developer mode`



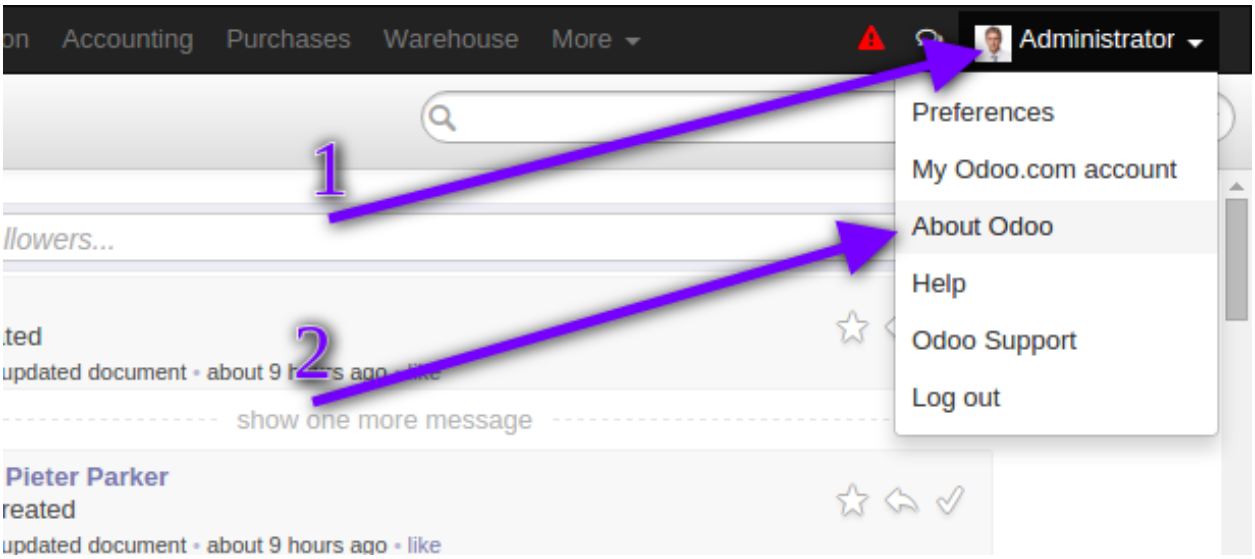
10.0+ with the web_debranding

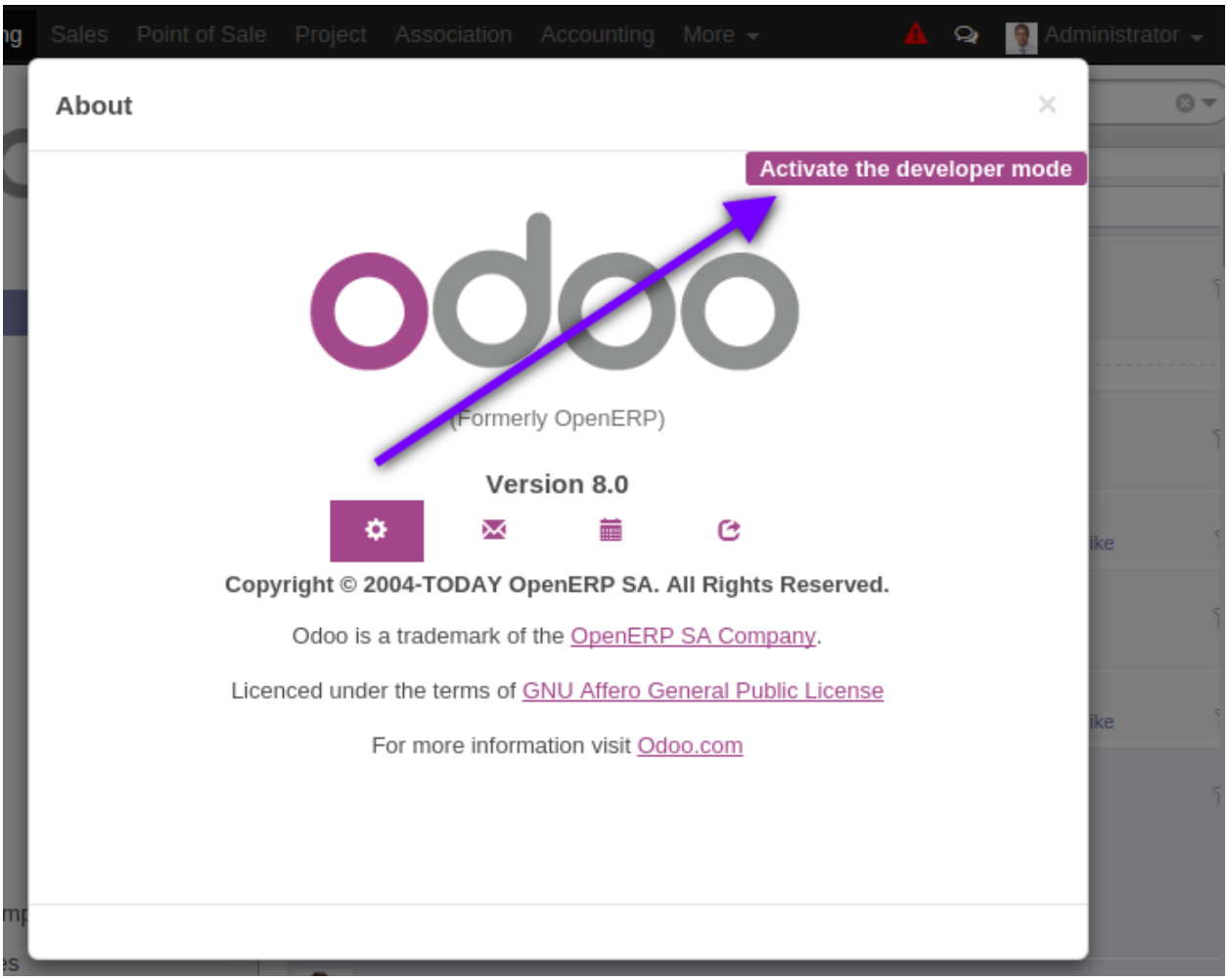
- go to the User menu on the upper right corner
- click `Developer mode`



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

CHAPTER 8

Odoo administration

Official docs:

- https://www.odoo.com/documentation/8.0/setup/install.html
- https://www.odoo.com/documentation/8.0/setup/deploy.html

Odoo installation

Contents

Note: This article covers development installation only. For production installation follow https://github.com/it-projects-llc/install-odoo

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=(
"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 demostration 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".
# 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 ODOO_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
```

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
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
# 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 )
```

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://;
        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;
        }
}
```

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 pip install gevent
    ```

  - odoo 10.0

    ```
    python -c "import gevent" || sudo pip install gevent
    python -c "import psycogreen" || sudo pip install psycogreen
    ```

- 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

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.

`--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

Longpolling

Hidden feature of Multiprocessing is automatic run gevent process for longpolling support.

Longpolling is an extra proccess, i.e. if you have `--workers=2` then you will get 2 worker processes and 1 gevent process

`--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␣
↪posgresql")
```

More accurate explanantion of this option is as following:

> `db_maxconn` – specify the the maximum number of physical connections to posgresql **per odoo process**

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␣
↪occured, 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 posgresql
  - 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.

--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")
```

--addons-path

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 beggining of `addons_path` list has more priority.

--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
```

Log levels

- CRITICAL
- ERROR
- WARNING
- INFO
- DEBUG
- NOTSET

--load

TODO

PosBox

Official docs:

- https://www.odoo.com/documentation/user/9.0/point_of_sale/overview/setup.html

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` 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 -e POSTGRES_USER=odoo -e POSTGRES_PASSWORD=odoo --name db-posbox-8.0␣
↪postgres:9.5

docker run \
-p 9069:8069 \
-p 9072:8072 \
-e ODOO_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
```

Source of this docker can be found here: https://github.com/it-projects-llc/install-odoo/tree/8.0/dockers/posbox

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

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.

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
```

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
```

How to edit odoo source

Comment out line 354 in `hw_escpos/controllers/main.py`

```
nano /home/pi/odoo/addons/hw_escpos/controllets/main.py
```

i.e. replace `driver.push_task('printstatus')` with

```
# driver.push_task('printstatus')
```

sync and reboot posbox

```
sync
reboot
```

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 9

Continuous Delivery

TODO

CHAPTER 10

IDE

Emacs

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

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
- pyhton
- 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). Install it by

```
sudo pip install pylint
```

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

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")'
```

PyCharm

PyCharm

Remote access with pgAdmin to Odoo postgre database on Ubuntu

This is for PgAdmin integration, but same method working with PyCharm.

STEP #1 – get pgAdmin Install pgAdmin from pgadmin.org

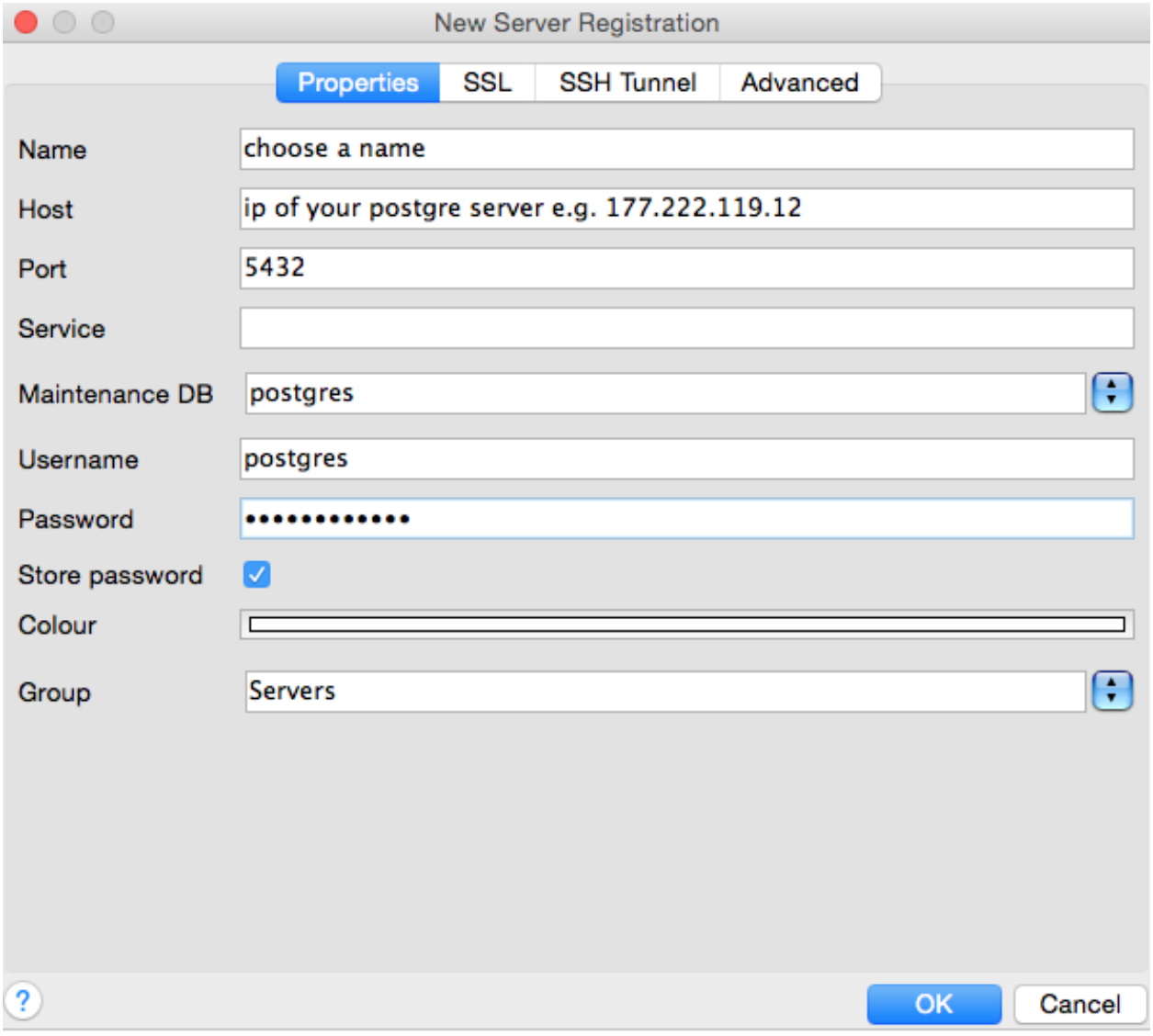
STEP #2 – allow postgre server remote connections from everywhere Open etc/postgresql/9.x/main/pg_hba.conf and add following line: host all all all md5

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/

Tmux

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/ugrade-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

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

# 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'
```

Visual Studio Code

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

- Fallow the same instructions in (emacs-pylint) to install pylint and Pylint Odoo plugin. Then make same configurations in pylintrc file as descriped there.

Attention: pylintrc file can be placed in the user invirument to work for all projects. like for debian "~/.pylintrc"

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 autocompleate/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 sow 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.

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 databace, config or user name and password.

sorce

CHAPTER 11

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

Usage

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". Simply add following lines to `~/.ssh/config` file 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

Edit files on remote server

sshfs

```
sudo apt-get install sshfs
mkdir /mnt/remote-files/

# use your HOST and PORT
# Warnings. It's not secure to use such mounting for production server.
sshfs -o allow_other,IdentityFile=~/.ssh/id_rsa root@HOST:/root/odoo /mnt/remote-
↪files -p PORT
# update /etc/fuse.conf if needed

# to unmount use
sudo umount /mnt/remote-files
```

win-sshfs (Windows)

After launching the win-sshfs program, you will be presented with a graphical interface to make the process of mounting a remote file share simple.

Step 1: Click the Add button in the lower left corner of the window.

Step 2: Enter a name for the file share in the Drive Name field.

Step 3. Enter the IP of your droplet in the Host field.

Step 4. Enter your SSH port. (Leave as port 22 unless you have changed the SSH port manually).

Step 5. Enter your username in the Username field. (Unless you have set up user accounts manually you will enter root in this field).

Step 6. Enter your SSH password in the password field. (Note on Windows you will need to have your droplet configured for password logins rather than ssh-key-authentication).

Step 7. Enter your desired mount point in the Directory field. (Enter / to mount the file system from root. Likewise you can enter /var/www or ~/ for your home directory).

Step 8. Select the drive letter you would like Windows to use for your droplets file system.

Step 9. Click the Mount button to connect to the droplet and mount the file system.

Now your virtual server's file system will be available through My Computer as the drive letter you chose in step 8.

References

- https://www.digitalocean.com/community/tutorials/how-to-use-sshfs-to-mount-remote-file-systems-over-ssh
- https://askubuntu.com/questions/780705/fuse-unknown-option-defer-permissions

Deploying x2go server

x2go allows you to run remotely browser (or any other application on x-server)

- Start x2go server on 2222 port

source: https://hub.docker.com/r/paimpozhil/docker-x2go-xubuntu/

```
docker run --name x2go -p 2222:22 -t -d paimpozhil/docker-x2go-xubuntu
docker logs x2go
```

- note the root/dockerx passwords
- Connect to your server using port forwarding (`-L` option), e.g.

```
ssh -L 2222:localhost:2222 user@server.example.com
```

- port `2222` is available now on your localhost, connect to it using *x2go client*

X2GO Client

- *Run or start x2go server container*
- 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

```
Host : localhost
Port : 2222
Username : dockerx
Password : (get it from the Docker logs when starting the server container)
```

Containers administration

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


############################
# 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="10.0.3.123"  # use one from network subnet
PORT="10100"  # unique per each developer

lxc init ubuntu-daily:16.04 ${CONTAINER} -p default -p docker
lxc network attach ${LXD_NETWORK} ${CONTAINER} eth0
lxc config device set ${CONTAINER} eth0 ipv4.address ${LOCAL_IP}
lxc config set ${CONTAINER} security.privileged true

# forward ssh port
iptables -t nat -A PREROUTING -p tcp -i eth0 --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

lxc start ${CONTAINER}
lxc exec ${CONTAINER} -- mkdir -p /root/.ssh
lxc exec ${CONTAINER} -- bash -c "curl --silent https://github.com/${GITHUB_USERNAME}.
↪keys >> /root/.ssh/authorized_keys"
# 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 update
lxc exec  ${CONTAINER} -- apt dist-upgrade -y
lxc exec  ${CONTAINER} -- apt install docker.io htop nginx -y

## nginx on host machine
cd /tmp/
curl -s https://raw.githubusercontent.com/it-projects-llc/odoo-development/master/
↪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 12

Other

RST format

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 distict 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 finaly you can enter your code.

Also you can use `inplace code reference` by using ‘‘ ‘‘.

Selection

- `**bold**`
- `*italic*`
- ``` ``code`` ```

Lists

- * - not numerated
- #. - numerated
- 1,2,3, ... - numerated

Links

- internal link:

```
:doc:`Link Text<../relative/path/to/article>`
```

- external link:

```
`Link Text <https://google.com/>`_
```

More documentations

- http://docutils.sourceforge.net/docs/user/rst/quickref.html

Adjust chromium window size script

You can make screenshot with size exactly you need.

Open chromium. Do not expand window (or in wount 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.

Want to talk with other developers?

Check out our telegram group, but don't take it too seriously ;-)

Need our service?

For module development contact us by email or fill out request form:

- it@it-projects.info
- https://www.it-projects.info/page/website.contactus
