Odoo development Documentation

Выпуск master

IT-Projects LLC

апр. 25, 2022

Оглавление

Help us maintain these docs up-to-date

Spread the word about following pages:

- https://odoo-debranding.com: Odoo Debranding modules + useful links for developers
- https://odoo-debranding.com/odoo-ce-vs-ee/: Odoo Editions Comparison
- https://odoo-debranding.com/oca/: List of OCA repositories

Глава 1

First steps

- Install odoo
- take the course Building a module
- read the article Source diving
- Get tasks from your Guru!
- Fork repo, clone repo to you machine, make commits, push updates, create Pull Request

Глава 2

Разработка модуля

2.1 Документы и манифесты

2.1.1 файлы

** Все ** файлы из этого раздела должны быть полностью [1] _ подготовлены ** перед ** любыми другими файлами в новом модуле. Это поможет вам пересмотреть требования еще раз, прежде чем начать.

README.rst

README OCA

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

README IT-проектов

- https://gitlab.com/itpp/handbook/blob/master/technical-docs/README.rst.md

док / index.rst

- Это описание будет доступно в магазине приложений на вкладке * Документация *. Пример: https://www.odoo.com/apps/modules/8.0/pos_multi_session/
- Не использует OCA
- “ *Doc / index.rst‘* ИТ-проектов доступен здесь: https://gitlab.com/itpp/handbook/blob/master/technical-docs/usage-instruction.md

__manifest__.py (__openerp__.py)

Манифест OCA

https://github.com/OCA/maintainer-tools/blob/master/template/module/__openerp__.py

имя

Это должно быть нетехническое название модуля

резюме

Краткое описание модуля. Например, вы можете описать, какая проблема решается модулем. Это может звучать как лозунг.

категория

Категории из списка ниже являются предпочтительными.

- “ Accounting“
- “ Discuss“
- “ Управление документами“
- “ eCommerce“
- “ Человеческие ресурсы“
- “ Industries“
- “ Localization“
- “ Manufacturing“
- “ Marketing“
- “ Точка продажи“
- “ Productivity“
- “ Project“
- “ Purchases“
- “ Sales“
- “ Warehouse“
- “ Website“
- “ Дополнительные инструменты“

скрытый

Для технических модулей может быть использована категория "Hidden":

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

Такие модули исключены из результатов поиска в магазине приложений.

версия

версия в IT-проектах

https://gitlab.com/itpp/handbook/blob/master/technical-docs/__manifest__.md#version

версия в OCA

https://github.com/OCA/odoo-community.org/blob/master/website/Contribution/CONTRIBUTING.rst#version-numbers

автор

Сначала используйте компанию, а затем разработчика (ов):

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

В основном, если модуль уже существует и вы делаете небольшие обновления исправления, вы не должны добавлять свое имя авторам.

автор в OCA

https://github.com/OCA/odoo-community.org/blob/master/website/Contribution/CONTRIBUTING.rst#backporting-odoo-modules

интернет сайт

Ссылка на персональную страницу на веб-сайте компании (например, “ "https://it-projects.info/team/yelizariev" “)

лицензия

ООО «ИТ-Проекты» использует следующие лицензии:

- “ "GPL-3" “ для odoo 8.0 и ниже
- “ "LGPL-3" “ для odoo 9.0 и выше

Для репозиториев OCA используйте “ "AGPL-3" “.

external_dependencies

Проверьте, существует ли какая-либо библиотека python:

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

Проверьте, существует ли какое-либо системное приложение:

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

Смотрите также: *External dependencies in odoo*

Шаблон IT-проектов

- https://gitlab.com/itpp/handbook/blob/master/technical-docs/__manifest__.md

и эти два файла:

changelog.rst *<https://gitlab.com/itpp/handbook/blob/master/technical-docs/changelog.rst.md>*‘__ icon.png <https://gitlab.com/itpp/handbook/blob/master/technical-docs/icon.png.md>‘__

2.2 Методические рекомендации

Источник:

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

2.2.1 Комментарии

Прежде всего, комментарии в источнике обязательны, если неясно ** почему ** что-то делают.

Кроме того, вы можете добавить комментарии о **, что ** вы делаете, если это может быть полезно.

2.3 Odoo Python

2.3.1 Python декораторы

Оригинальная статья

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

@ api.one

Предупреждение: декоратор устарел. Используйте взамен “ @ api.multi“

api.one предназначен для использования, когда метод вызывается только для одной записи. Это обеспечивает отсутствие множественных записей при вызове метода с помощью декоратора api.one. Допустим, у вас есть запись партнера = res.partner (1,). Это только одна запись, и есть метод, например (в res.partner)

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

называть это так работает

```
partner.get_name()
```

Но если будет больше записей, например:

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

вызов этого вызовет предупреждение, сообщающее, что вы можете вызывать его только на одной записи.

@ api.multi

Примечание: Методы без декораторов работают так же, как “ @ api.multi“

что нибудь. Например:

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

@ api.model

И api.model считается используемым, когда вам нужно что-то сделать с самой моделью, и вам не нужно изменять / проверять какие-то точные записи / записи модели. Например, может существовать метод, который возвращает некоторую метаинформацию о структуре модели или некоторые вспомогательные методы и т. Д. Также в документации говорится, что этот API хорошо использовать при переходе от старого API, поскольку он «вежливо» преобразует код в новый API. , Также по моему опыту, если вам нужен метод, чтобы вернуть что-то, модель декоратор подходит для этого. api.one возвращает пустой список, поэтому это может привести к неожиданному поведению при использовании метода api.one on, когда он должен что-то возвращать.

2.3.2 Чистый Питон

Сравните два массива

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

2.3.3 res.config.settings

Based on https://github.com/odoo/odoo/blob/10.0/odoo/addons/base/res/res_config.py

“ res.config.settings“ - мастер базовой конфигурации для настроек приложения. Он обеспечивает поддержку установки значений по умолчанию, назначения групп пользователям-сотрудникам и установки модулей. Чтобы создать такой мастер настройки, определите модель, например:

```
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(...)
```

Метод “ execute“ (кнопка * Apply *) обеспечивает некоторую поддержку, основанную на соглашении об именах:

- Для поля типа “ default_XXX“, “ execute“ устанавливает (глобальное) значение по умолчанию поля “ XXX“ в модели, названной “ default_model“, равным значению поля.
- Для логического поля, такого как “ group_XXX“, “ execute“ добавляет / удаляет ’implied_group’ в / из подразумеваемых групп ’group’, в зависимости от значения поля. По умолчанию «группа» - это группа «Сотрудник». Группы задаются своим идентификатором xml. Атрибут group может содержать несколько идентификаторов xml, разделенных запятыми.
- Для логического поля, подобного “ module_XXX“, “ execute“ запускает немедленную установку модуля с именем “ XXX“, если поле имеет значение “ True“.
- Для других полей метод “ execute“ вызывает все методы с именем, которое начинается с “ set_“; такие методы могут быть определены для реализации эффекта этих полей.

Метод “ default_get“ извлекает значения, отражающие текущее состояние полей, таких как “ default_XXX“, “ group_XXX“ и “ module_XXX“. Он также вызывает все методы с именем, которое начинается с “ get_default_“; такие методы могут быть определены для предоставления текущих значений для других полей.

пример

```
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
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    def set_params(self):
        self.ensure_one()

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

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

2.3.4 Обновление настроек при установке модуля

Чтобы обновить настройки из любого “ res.config.settings“, выполните следующие действия:

default_XXX

СДЕЛАТЬ

group_XXX

Добавьте ** подразумеваемую группу (ы) ** к ** группе ** через поле “ implied_ids“

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

module_XXX

Добавьте XXX к параметру зависимость в файле *__manifest__.py*.

Другие поля

Обычно другие поля сохраняются в “ ir.config_parameter“, поэтому просто *update ir.config_parameter*, например

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

2.3.5 Веб контроллеры

Отправить значения на веб-страницу

Если вам нужно передать на страницу рендеринга некоторые переменные, вам нужно поместить эти переменные в словарь и поместить их в качестве второго аргумента:

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

2.3.6 One2one поле в odoo

Odoo ORM не поддерживает поля “ One2one“, но вы можете сделать их вручную. В приведенном ниже примере мы устанавливаем связь one2one между моделями “ fleet.vehicle“ и “ account.asset.asset“.

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

```
class Fleet(models.Model):

_inherit = 'fleet.vehicle'
...
asset_id = fields.Many2one('account.asset.asset', compute='compute_asset', inverse='asset_inverse')
asset_ids = fields.One2many('account.asset.asset', 'vehicle_id')

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

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


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

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

TODO: заменить “ @ api.one“ на “ @ api.multi“

2.3.7 заполнение множества значений

Чтобы заполнить или манипулировать полем one2many или many2many соответствующими значениями (записями), вам необходимо использовать специальную команду, как указано ниже.

Odoo 15.0+

First import `fields`

```
from odoo import fields
# or Command directly:
# from odoo.fields import Command
```

Then assign list of following commands to a x2many field:

- `fields.Command.create(values)`
- `fields.Command.update(id, values)`
- `fields.Command.delete(id)`
- `fields.Command.unlink(id)`
- `fields.Command.link(id)`
- `fields.Command.clear()`
- `fields.Command.set(ids)`

Based on https://github.com/odoo/odoo/blob/84f89d6ff887e750ea79656328362333cfce27fd/odoo/fields.py#L2868-L2982

Odoo 14.0-

Этот формат представляет собой список триплетов, выполняемых последовательно, где каждый триплет является командой для выполнения на множестве записей. Не все команды применяются во всех ситуациях. Возможные команды:

- ** (0, _, значения) ** добавляет новую запись, созданную из предоставленного ** значения ** dict.
- ** (1, id, значения) ** обновляет существующую запись id ** id ** значениями в ** значениях **. Не может использоваться в ~ *.create*.
- ** (2, id, _) ** удаляет запись id ** id ** из набора, а затем удаляет ее (из базы данных). Не может использоваться в ~ *.create*.
- ** (3, id, _) ** удаляет запись id ** id ** из набора, но не удаляет ее. Не может использоваться на ~ *openerp.fields.One2many*. Не может использоваться в ~ *.create*.
- ** (4, id, _) ** добавляет существующую запись id ** id ** к набору. Не может использоваться на ~ *openerp.fields.One2many*.
- ** (5, _, _) ** удаляет все записи из набора, что эквивалентно явному использованию команды ** 3 ** для каждой записи. Не может использоваться на ~ *openerp.fields.One2many*. Не может использоваться в ~ *.create*.
- ** (6, _, ids) ** заменяет все существующие записи в наборе на список ** ids **, что эквивалентно использованию команды ** 5 **, за которой следует команда ** 4 ** для каждого ** id ** in ** id **. Не может использоваться на ~ *openerp.fields.One2many*.

Примечание: Значения, помеченные как ** _ ** в приведенном выше списке, игнорируются и могут быть чем угодно, обычно ** 0 ** или ** False **.

Based on https://github.com/odoo/odoo/blob/14.0/odoo/models.py

2.3.8 поля

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

- *Полевое наследование*
- *Типы полей*
  - *логический*
  - *голец*
  - *Текст*
  - *HTML*
  - *целое число*
  - *терка*
  - *Дата*
  - *DateTime*
  - *двоичный*
  - *выбор*
  - *Ссылка*
  - *Many2one*
  - *One2many*
  - *Many2many*
- *Конфликты имен*
- *Поля по умолчанию*
- *Вычисляемые поля*
- *обратный*
- *Мульти поля*
- *Связанное поле*
- *Поле собственности*
- *WIP копируемая опция*
- *Специальные поля*
  - *активный*

Теперь поля являются свойством класса:

```
from openerp import models, fields


class AModel(models.Model):

    _name = 'a_name'
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    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
```

Полевое наследование

Одна из новых функций API - возможность изменять только один атрибут поля:

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

Типы полей

логический

Поле логического типа:

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

голец

Сохранить строку с переменной len .:

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

Конкретные варианты:

- размер: данные будут обрезаны до указанного размера
- перевести: поле можно перевести

Текст

Используется для хранения длинного текста.

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

Конкретные варианты:

- перевести: поле можно перевести

HTML

Используется для хранения HTML, предоставляет виджет HTML .:

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

Конкретные варианты:

- перевести: поле можно перевести

целое число

Сохраните целочисленное значение. Нет поддержки значения NULL. Если значение не установлено, возвращается 0:

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

терка

Хранить значение с плавающей запятой. Нет поддержки значения NULL. Если значение не установлено, возвращается 0.0. Если задана опция цифр, будет использоваться числовой тип:

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

Конкретные варианты:

- цифры: принудительное использование числового типа в базе данных. Параметр может быть кортежем (int len, float len) или вызываемым, который возвращает кортеж и принимает курсор в качестве параметра

Дата

Дата магазина. Поле предоставляет несколько помощников:

- “ context_today“ возвращает строку даты текущего дня на основе tz
- “ today“ возвращает строку текущей системной даты
- “ from_string“ возвращает datetime.date () из строки
- “ to_string“ возвращает строку даты из datetime.date

dummy literal

```
>>> from openerp import fields

>>> adate = fields.Date()
>>> fields.Date.today()
```

(continues on next page)

(продолжение с предыдущей страницы)

```
'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

Хранить дату и время Поле предоставляет некоторый помощник:

- “ context_timestamp“ возвращает строку даты текущего дня на основе tz
- “ now“ возвращает строку текущей системной даты
- “ from_string“ возвращает datetime.date () из строки
- “ to_string“ возвращает строку даты из datetime.date

dummy literal

```
>>> 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'
```

двоичный

Сохраните файл, закодированный в base64, в столбце bytea:

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

выбор

Сохраните текст в базе данных, но предложите виджет выбора. Это не вызывает никаких ограничений выбора в базе данных. Выбор должен быть установлен в виде списка кортежей или вызываемого элемента, который возвращает список кортежей:

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

Конкретные варианты:

- selection: список кортежей или вызываемых имен, которые принимают набор записей в качестве входных данных

- size: опция size = 1 обязательна при использовании целых индексов, а не строк

При расширении модели, если вы хотите добавить возможные значения в поле выбора, вы можете использовать аргумент ключевого слова *selection_ add*

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

Since Odoo 14.0 you have to specify ondelete attribute.

`ondelete` **provides a fallback mechanism for any overridden** field with a selection_add. It is a dict that maps every option from the selection_add to a fallback action. This fallback action will be applied to all records whose selection_add option maps to it. The actions can be any of the following:

- `'set null'` – the default, all records with this option will have their selection value set to False.
- `'cascade'` – all records with this option will be deleted along with the option itself.
- `'set default'` – all records with this option will be set to the default of the field definition
- `<callable>` – a callable whose first and only argument will be the set of records containing the specified Selection option, for custom processing. e.g.:

```
my_selection = fields.Selection(selection_add=[
     ('bacon', "Bacon"),
], ondelete={'bacon': lambda records: record.write({'my_selection': 'bar'})})
```

Ссылка

Сохраните произвольную ссылку на модель и строку:

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

Конкретные варианты:

- selection: список кортежей или вызываемых имен, которые принимают набор записей в качестве входных данных

Many2one

Сохранить отношение к совместной модели:

```
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)
```

Конкретные варианты:

- comodel_name: имя противоположной модели
- делегат: установите его в “ True“, чтобы сделать поля целевой модели доступными из текущей модели (соответствует “ _inherits“)

One2many

Сохранить отношение ко многим строкам совместной модели:

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

Конкретные варианты:

- comodel_name: имя противоположной модели
- inverse_name: реляционный столбец противоположной модели

Many2many

Сохраните отношение ко многим многим строкам совместной модели:

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

Конкретные варианты:

- comodel_name: имя противоположной модели
- отношение: имя реляционной таблицы
- columns1: имя левого столбца реляционной таблицы (ссылка на запись в текущей таблице)
- columns2: имя правого столбца реляционной таблицы (ссылка на запись в таблице * comodel_name *)

Чтобы сделать два взаимных много-много-много полей в разных моделях, используйте в них одну и ту же таблицу отношений и обратные столбцы:

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

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

Конфликты имен

Примечание: поля и имя метода могут конфликтовать.

Когда вы называете запись диктовкой, она заставляет смотреть на столбцы.

Поля по умолчанию

По умолчанию теперь используется ключевое слово поля:

Вы можете приписать ему значение или функцию

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

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

Использование веселья заставит вас определить функцию перед определением полей.

Заметка. Значение по умолчанию не может зависеть от значений других полей записи, т. Е. Вы не можете читать другие поля через “ self“ в функции.

Вычисляемые поля

Больше нет прямого создания полей.

Вместо этого вы добавляете “ compute“ kwarg. Значение - это имя функции в виде строки или функции. Это позволяет иметь определение полей поверх класса:

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

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

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

Функция может быть недействительной. Он должен изменить свойство записи для записи в кеш:

```
self.name = new_value
```

Имейте в виду, что это назначение приведет к записи в базу данных. Если вам нужно сделать массовое изменение или вы должны быть осторожны с производительностью, вы должны сделать классический вызов, чтобы написать

Чтобы обеспечить функцию поиска для несохраненного вычисляемого поля, вы должны добавить к полю “ search“. Значение - это имя функции в виде строки или ссылка на ранее определенный метод. Функция принимает второго и третьего члена доменного кортежа и возвращает сам домен:

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

обратный

Обратный ключ позволяет вызвать вызов декорированной функции, когда поле написано / «создано»

Мульти поля

Чтобы иметь одну функцию, которая вычисляет несколько значений:

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

Связанное поле

Больше нет полей “ fields.related“.

Вместо этого вы просто устанавливаете аргумент name, связанный с вашей моделью:

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

“ Type“ kwarg больше не нужен.

Установка “ store“ kwarg автоматически сохранит значение в базе данных. С новым API значение соответствующего поля будет автоматически обновлено, сладко.

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

Примечание: При обновлении любого связанного поля не все переводы связанного поля переводятся, если поле сохранено !!

Модификация связанных связанных цепочек вызовет аннулирование кэша для всех элементов цепочки.

Поле собственности

В некоторых случаях значение поля должно меняться в зависимости от текущей компании.

Чтобы активировать такое поведение, теперь вы можете использовать опцию *company_ dependent*.

Заметная эволюция в новом API заключается в том, что «поля свойств» теперь доступны для поиска.

WIP копируемая опция

Есть запущенный разработчик, который не позволит переопределить копию, просто установив параметр копирования в полях:

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

Специальные поля

активный

СДЕЛАТЬ

См. Https://github.com/odoo/odoo/blob/11.0/odoo/models.py#L3556-L3560.

2.3.9 Ограничения модели

Odoo предоставляет два способа настройки автоматически проверяемых инвариантов: *ограничения Python <openerp.api.constrains> ‘и* ограничения SQL <openerp.models.Model._sql_constraints> ‘.

Ограничение Python определяется как метод, украшенный ~ *openerp.api.constrains* и вызываемый для набора записей. Декоратор определяет, какие поля участвуют в ограничении, поэтому ограничение автоматически оценивается при изменении одного из них. Ожидается, что метод вызовет исключение, если его инвариант не выполняется:

```
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
```

Ограничения SQL определяются через атрибут модели ~ *openerp.models.Model._sql_constraints*. Последний присваивается списку троек строк ‘‘ (name, sql_definition, message) , где `` name - допустимое имя ограничения SQL, ‘‘ sql_definition‘‘ - выражение ‘‘ table_constraint_‘‘, и ‘‘ message‘‘ - это сообщение об ошибке.

2.3.10 Отчеты моделей через представления PostgreSQL

Postgres View это своего рода таблица, которая физически не материализуется. Вместо этого запрос выполняется каждый раз, когда на представление ссылаются в запросе.

Чтобы создать Postgres View в odoo, сделайте следующее:

- создать новую модель
- все поля должны иметь флаг ‘‘ readonly = True‘‘.
- задайте параметр ‘‘ _auto = False‘‘ для модели odoo, чтобы таблица, соответствующая полям, не создавалась автоматически.
- добавить метод ‘‘ init (self, cr) ‘‘, который создает представление PostgreSQL, соответствующее полям, объявленным в модели.
  - Поле ‘‘ id‘‘ должно быть указано в части ‘‘ SELECT‘‘. См пример ниже
- добавить взгляды на модель обычным способом

Пример:

```
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')
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    @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
        )""")
```

2.3.11 Внешние зависимости в odoo

какая

Внешние зависимости - это пакеты Python или любые двоичные файлы, которые должны быть установлены для работы модуля.

Как

В файлах python, где вы используете внешние зависимости, вам нужно добавить “ try-Кроме“ с журналом отладки.

```
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)
```

Это правило не применяется к тестовым файлам, поскольку эти файлы загружаются только при выполнении тестов, и в этом случае ваш модуль и его внешние зависимости установлены.

Также вам необходимо добавить внешние зависимости в *manifest*.

Почему

Odoo загружает Python-файлы модуля, когда выполняются следующие условия:

- модуль имеет статическую папку (например, для иконки)
- модуль, помеченный как устанавливаемый в *manifest*, т.е. модуль * можно * установить

Видно, что odoo загружает файлы Python, даже если модуль не установлен (и даже не предназначен для установки). Но модули обычно добавляются в addons-path как часть некоторого репозитория (например, * pos-addons *). Вот почему импорт внешних зависимостей без ‘‘ try-кроме‘‘ приводит к проблемам при добавлении репозитория в *addons-path*.

2.4 XML

2.4.1 Создать запись модели

Создать новую запись

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

Если модель существует, она будет изменена. Создание записи в модуле объявлено. Чтобы изменить модель, созданную в другом модуле, добавьте имя мула перед id

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

2.4.2 Xpath

Добавить некоторые атрибуты в узел

Код:

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

Qweb выражение

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

После рендеринга это становится обычным атрибутом:

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

Важный

Внутри

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

Вы можете поставить ** только ** “ <attribute name=“ and nothing more.

Для проверки xpath

Код:

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

2.4.3 Основные вещи

Вызовите метод некоторой модели и поместите результат в переменную

Код:

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

Здесь * веб-сайт * означает, что вы используете веб-сайт = True в контроллере. ТОДО, я могу ошибаться.

Получите значение некоторого параметра ir.config_parameter и поместите его в переменную

Код:

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

Показать значение переменной

Код:

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

Используйте переменную в условии

Код:

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

Получить переменную, передаваемую render () в шаблоне XML

Код:

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

my_var является элементом словаря 'values' (второй аргумент render ()).

2.4.4 Наследование

Столкновения и приоритет

Если два или более шаблонов xml наследуют один и тот же родительский шаблон, они могут иметь одинаковые приоритеты. Это может привести к конфликтам и неожиданному поведению. То, что вам нужно, это просто явно установить приоритет в вашем шаблоне:

```
<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>
```

Меньший приоритет означает предварительное исполнение.

Приоритет по умолчанию - 16.

2.4.5 “ Group_id“ в представлениях

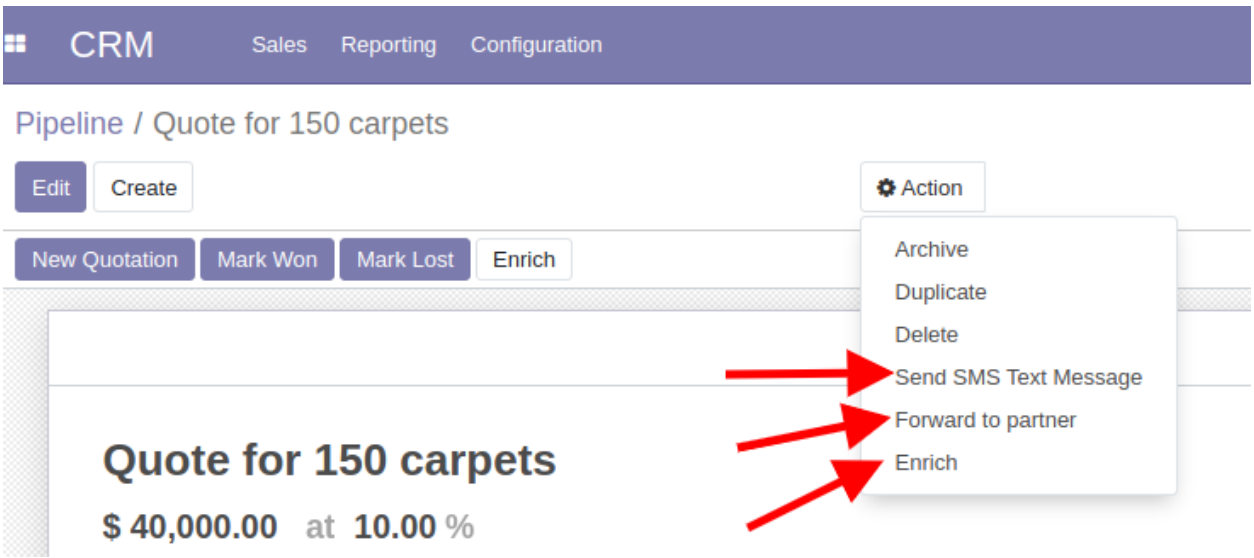
“ groups_id“ в представлениях (в основном в представлениях, которые наследуют другие представления) позволяет указать, для какой группы производится это наследование.

Например, если вам нужно добавить кнопку, которая будет доступна только для менеджеров, сделайте следующее:

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

Если “ groups_id“ опущено, то обновление применяется ко всем пользователям.

2.4.6 Actions Menu



To add such button, you need to create `ir.actions.act_window` record with `binding_model_id` value.

14.0+

Example for Odoo 14.0+:

```
<record id="crm_lead_act_window_sms_composer_multi" model="ir.actions.act_window">
    <field name="name">Send SMS Text Message</field>
    <field name="res_model">sms.composer</field>
    <field name="view_mode">form</field>
    <field name="target">new</field>
    <field name="context">{
        'default_composition_mode': 'comment',
        'default_res_id': active_id,
    }</field>
    <field name="binding_model_id" ref="model_crm_lead"/>
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    <field name="binding_view_types">form</field>
</record>
```

See also https://www.odoo.com/documentation/master/howtos/backend.html#launching-wizards

13.0

Example for Odoo 13.0:

```
<act_window id="crm_lead_act_window_sms_composer_multi"
    name="Send SMS Text Message"
    binding_model="crm.lead"
    res_model="sms.composer"
    binding_views="form"
    view_mode="form"
    target="new"
    context="{
        'default_composition_mode': 'comment',
        'default_res_id': active_id,
    }"
/>
```

See also https://www.odoo.com/documentation/13.0/howtos/backend.html#launching-wizards

12.0-

Example for Odoo 12.0-:

- there is no `binding_views` option
- `src_model` is the same as `binding_model`

```
<act_window
    id="act_pos_config_sessions"
    name="Sessions"
    src_model="pos.config"
    res_model="pos.session"
    domain="[('config_id', '=', active_id)]" />
```

See also https://www.odoo.com/documentation/12.0/howtos/backend.html#launching-wizards

2.5 HTML

2.5.1 Активные элементы

Link-кнопка, которая вызывает контроллер

Код:

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

Здесь action = "/ shop / checkout" устанавливает адрес контроллера. Класс a-submit обычно означает «делать то, что в форме».

Отправить с помощью кнопки

Код:

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

При этом в контроллере в ** post будут доступны некоторые значения из исходной формы, такие как <input/> ,

2.6 CSS

2.6.1 CSS советы и хитрости

Добавьте свой CSS на шаблон

Код:

```
<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.

Скрыть поля

Скрыть все дочерние элементы (имеющие атрибут bill = '1') владельца класса oe_website_sale (имеющие атрибут bill_enabled = '0')

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

2.7 YAML

2.7.1 Чистый ЯМЛ

СДЕЛАТЬ

2.7.2 YAML в оду

СДЕЛАТЬ

2.8 Javascript

2.8.1 наследование

СДЕЛАТЬ

2.8.2 core.bus

core.bus (web.bus в 8.0) используется для обработки событий js между модулями.

использование

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

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

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

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

2.8.3 Удаленный вызов процедур (RPC)

Метод вызова

```
/**
 * 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,
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        method: method,
        args: args,
        kwargs: kwargs
    }, options);
},
```

Как вызвать метод мастера из js

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

2.9 Внешний интерфейс

2.9.1 веб-страница

** Общие **

Откройте новый проект:

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

Добавьте “ website“ как зависимость к “ newpage“:

```
'depends': '[website]'
```

затем добавьте флаг website = True на контроллер, это установит несколько новых переменных в объекте запроса и позволит использовать макет сайта в нашем шаблоне.

** Создание страниц **

** 1 способ **

Напишите следующий код в “ 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')
```

Новая веб-страница появится, добавив - “ / new-page / “ “ http.request.render (’newpage.index’) “ - загрузив тамплату для новой страницы

Шаблон “ templates.xml“

```
<openerp>
    <data>
        <templateid="index">
            <t t-call="website.layout">
                <t t-set="title">New page</t>
```

(continues on next page)

(продолжение с предыдущей страницы)

```
                    <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“ означает, что используются элементы шаблона сайта.

После перезапуска сервера при обновлении модуля (для обновления манифеста и шаблона) перейдите по адресу http: // localhost: 8069 / new-page /. Вы увидите новую страницу с заголовком * «Моя первая веб-страница» * и текстом * «Привет, мир!» *

2 пути

Напишите в шаблоне следующее:

```
<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" “ создает страницу следующим образом: http: // localhost: 8069 / page / services /

Если добавить в “ 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>
```

Этот код добавит ссылку в главное меню.

2.10 Торговая точка (POS)

2.10.1 Новый POS-модуль

Добавление файла “ js“ в POS

Добавление ** javascript файла ** открывает новый набор возможностей в Odoo.

Давайте рассмотрим пример модуля “ POS Debt & Credit notebook““: <https://github.com/it-projects-llc/pos-addons/blob/15a6853768a888bb7c3fbfd3690ce0cb7537ff3e/pos_debt_notebook/data.xml#L16-L20::>‘__

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

Функция “ odoo.define“

Официальный документ о теме здесь:

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

Пример:

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

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

Предупреждение: Вы не можете переименовать переменную “ require“.

Примечание: В одном файле может быть несколько * JS-модулей *, хотя рекомендуется помещать их в разные файлы

Примечание: Вы можете использовать любую строку в качестве имени модуля, но рекомендуется использовать “ <ODOO_MODULE> , <JS_MODULE> , например `` point_of_sale.popups

Возвращаемое значение

JS-модуль может возвращать значение. Это значение может быть использовано в других js-модулях (того же odoo-модуля или других).

Например:

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

Затем мы можем использовать “ define_screen“ следующим образом:

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

Примечание: Если вы не используете значение, возвращаемое другим js-модулем, вы все равно можете * импортировать js-module * (через require (....)), чтобы убедиться, что этот модуль загружен перед выполнением вашего модуля.

Асинхронные модули

Может случиться так, что модуль должен выполнить некоторую работу, прежде чем он будет готов. Например, он может сделать RPC для загрузки некоторых данных. В этом случае модуль может просто вернуть отложенное (обещание). В этом случае система модулей просто будет ждать завершения отсроченного перед регистрацией модуля.

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

наследование

POS имеет два типа классов: модели, виджеты. Расширение этих классов немного отличается.

Примечание: Не у всех классов есть простой способ заставить их наследовать. Некоторые трюки доступны здесь <https://odoo-development.readthedocs.io/en/latest/dev/pos/gui.html>‘__

модель

Классы моделей работают только с данными и не работают с пользовательским интерфейсом напрямую.

Чтобы расширить класс такого типа, вам нужно использовать метод “ extend“. Создает копию класса с переопределенным методом. Обычно вам нужно переопределить исходный класс обновленным. Также, чтобы вызвать оригинальный метод, поместите оригинальный класс в переменную.

Вот пример <https://github.com/it-projects-llc/pos-addons/blob/fb8b072/pos_debt_notebook/static/src/js/pos.js#L23-L33>‘__:

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

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

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

Виджет

Классы виджетов работают с пользовательским интерфейсом.

Расширение виджета намного проще, чем расширение модели: просто используйте “ include“ и “ _super“.

Вот пример <https://github.com/it-projects-llc/pos-addons/blob/fb8b072/pos_debt_notebook/static/src/js/pos.js#L379-L385>‘__:

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

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

2.10.2 Расширение интерфейса

Кнопки действий

“ Action Buttons“ - это следующие кнопки:

- “ Note“
- “ Transfer“
- “ Guests“
- “ Bill“
- “ Split“
- “ Order“
- “ Discount“
- и т.п.

которые расположены выше ** Numpad **.

- Эти кнопки отображаются только после установки модуля ** pos_discount ** (кнопка “ Скидка“, позволяющая определить размер скидки на заказ) и модуля ** pos_restaurant ** (“ Split“, “ Гости‘) ‘кнопки и т. д.)
- Вы можете создавать свои собственные кнопки, назначая их действиям (например, ** open_popup **, ** screen ** и т. Д.).
- Чтобы создать кнопку, вам нужно унаследовать ** ActionButtonWidget ** Class, выбрать ** define_action_button ** и выбрать необходимое действие после нажатия соответствующей кнопки.

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

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

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

//declare a new variable and inherit ActionButtonWidget

var PopupButton = screens.ActionButtonWidget.extend({
  /*
  Thus PopupButton contains all methods from ActionButtonWidget.
  Now we need to define Template for our button,
  where the type of button you can find in Qweb (see below)
  */

template: 'PopupButton',
  /*
  We also need to choose the Action,
  which which will be executed after we click the button.
  For this purpose we define button_click method, where
  where name - Button name; widget - Button object;
  condition - Condition, which calls the button to show up
```

(continues on next page)

(продолжение с предыдущей страницы)

```
  (in our case, setting on show_popup_button option in POS config).
  */

button_click: function () {
  this.gui.show_popup('confirm', {
    'title': 'Popup',
    'body': 'Opening popup after clicking on the button',
    });
    }
  });

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

Определение “ template“ в “ Qweb“:

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

Для конкретного примера проверьте модуль ** POS Orders History ** ‘ <https://github.com/it-projects-llc/pos-addons/blob/12.0/pos_orders_history/static/src/js/screens.js#L22>‘__, где вы видите, что добавлена кнопка с надписью * История заказов *.

Создание пользовательских всплывающих окон

Используйте пользовательские всплывающие окна, чтобы предоставить информацию или предложить пользователям сделать что-то в POS. Вы можете определить внешний вид всплывающего окна.

** Давайте возьмем пример создания всплывающего окна ** “ Сканирования QR-кода в POS“‘ <https://github.com/it-projects-llc/pos-addons/blob/6eaac4e168d7cf854d302b298b068e2b38db822c/pos_qr_scan/static/src/js/qr_scan.js>‘__, где нам нужно было создать всплывающее окно, чтобы показать видео с камеры для сканирования QR-кодов.

Сначала приложите необходимые требования: <https://github.com/it-projects-llc/pos-addons/blob/6eaac4e168d7cf854d302b298b068e2b38db822c/pos_qr_scan/static/src/js/qr_scan.js#L10::>‘__

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

Затем создайте новый экземпляр, новое всплывающее расширение “ по умолчанию: <https://github.com/it-projects-llc/pos-addons/blob/6eaac4e168d7cf854d302b298b068e2b38db822c/pos_qr_scan/static/src/js/qr_scan.js#L29-L30::>‘__

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

Чтобы сделать всплывающее окно доступным обычными методами после объявления “ QrScanPopupWidget“, сделайте следующее: <https://github.com/it-projects-llc/pos-addons/blob/6eaac4e168d7cf854d302b298b068e2b38db822c/pos_qr_scan/static/src/js/qr_scan.js#L194::>‘__

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

поэтому его можно вызвать с помощью следующего кода: <https://github.com/it-projects-llc/pos-addons/blob/6eaac4e168d7cf854d302b298b068e2b38db822c/pos_qr_scan/static/src/js/qr_scan.js#L17-L20::>‘__

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

Пользовательские экраны

Список ** партнеров **, ** экран оплаты ** и ** экран экрана ** являются примерами “ экранов“.

Мы рассмотрим * пример создания пользовательского интерфейса *.

Для создания нового “ custom screen“ мы подключаем “ screen“ и “ gui“:

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

Затем мы объявляем новую переменную и наследуем “ ScreenWidget“:

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

Теперь “ CustomScreenWidget“ состоит из всех методов из “ ScreenWidget“. Затем нам нужно определить шаблон, где структура экрана описывается с помощью “ Qweb“:

```
template: 'CustomScreenWidget'
```

В “ Qweb“ мы определяем шаблон следующим образом:

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

(continues on next page)

(продолжение с предыдущей страницы)

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

Определите стили в файле “ css“, который вам нужен для экрана.

Этот “ Qweb“ будет отображаться каждый раз, когда запускается метод “ renderElement“ (до загрузки POS все экраны уже прорисованы и скрыты). Этот метод можно переопределить и, например, использовать для действий кнопок “ назад“ и “ next“:

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

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

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

Все экраны по умолчанию скрыты (кроме тех, которые вызываются после загрузки POS). Для того, чтобы открыть пользовательские экраны, необходимо определить их в списке экранов:

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

Чтобы открыть пользовательский экран, необходимо вызвать следующую функцию (например, после нажатия кнопки «Действие»):

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

where `this` is a pointer to `PosModel`.

Для конкретного примера проверьте модуль ** POS Orders History ** ‘ <https://github.com/it-projects-llc/pos-addons/blob/12.0/pos_orders_history/static/src/js/screens.js#L311>‘__, где‘ *orders_history_screen‘* определено.

2.10.3 Квитанции и принтеры

Таможенная квитанция

В POS есть два типа чеков:

1. “ PosTicket“ - отображается на экране после оплаты;
2. “ XmlReceipt“ - печатает в PosBox после нажатия кнопки Распечатать квитанцию на экране оплаты, если PosBox подключен к принтеру ESC POC.

Эти два чека реализованы в “ Qweb“ и генерируются после заказа на покупку. Если “ PosTicket“ разрешает любой дизайн в “ Qweb“, то для “ XmlReceipt“ вы можете использовать только строго определенные теги, которые поддерживаются принтером ESC POC.

Используя механизм “ t-extend“, который принимает имя шаблона для изменения в качестве параметра, вы можете расширить существующие шаблоны “ Qweb“ для получения. После этого может быть выполнена модификация с любым количеством поддиректив t-jquery.

Директивы “ t-jquery“ используют селектор CSS. Этот селектор используется в расширенном шаблоне для выбора узлов контекста, для которых может быть применена “ t-операция“. Если вы хотите добавить, например, другой заголовок для продукта в “ XmlReceipt“, вам нужно создать “ Qweb“ со следующим содержимым:

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

Для того же действия для PosTicket:

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

Одним из сложных, но в то же время гибких методов создания квитанции Клиента является загрузка поля с Сервера, где в текстовой форме описан шаблон “ Qweb“. После загрузки и перед печатью этот текстовый шаблон необходимо преобразовать в «формат XML» и получить данные на основе этого шаблона.

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

В POS вам нужно преобразовать этот текстовый формат в “ XML“ и сгенерировать квитанцию, используя этот шаблон:

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

Использование этого шаблона вместо стандартных требует генерации полученного “ XML“, для этого вам необходимо подключить “ Qweb“:

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

Чтобы определить пользовательскую функцию, которая будет генерировать “ Qweb“ пользователя следующим образом:

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

(continues on next page)

(продолжение с предыдущей страницы)

```
  return Qweb._render(template_name, options);
},
```

Эту функцию необходимо вызывать каждый раз, когда генерируется квитанция.

2.10.4 Дополнительные данные заказа

Загрузка данных в POS

По умолчанию POS загружает следующие модели:

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

Если мы добавили новое поле в бэкэнд и хотим, чтобы они были представлены в POS, мы можем использовать метод ** load_fields ** внутри “ PosModel“ ** ** инициализирующей функции **.

В следующем примере, взятом из модуля “ POS Debt & Credit notebook“, мы добавим несколько новых полей в модель “ account.journal““: <https://github.com/it-projects-llc/pos-addons/blob/fb8b0724fd4b5a0e66a64ece17643025e45330a8/pos_debt_notebook/static/src/js/pos.js#L29-L30::>‘__

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

Для загрузки новой модели в POS мы используем “ load_models (models, options) “. Описание взято из odoo

Загружает “ openerp“ модели в точке стартапа.

“ load_models“ принимает массив объявлений загрузчика модели.

Модели будут загружены в порядке массива. Если имя модели “ openerp“ не указано, данные сервера не будут загружены, но систему можно использовать для предварительной обработки данных перед загрузкой.

Аргументы загрузчика могут быть функциями, которые возвращают динамическое значение. Функция принимает “ PosModel“ в качестве первого аргумента и временный объект, который используется всеми моделями и может использоваться для хранения переходной информации между загрузками моделей.

Нет управления зависимостями. Модели должны быть загружены в правильном порядке. Недавно добавленные модели загружаются в конце, но параметры «после» и «до» могут использоваться для загрузки непосредственно перед / после другой модели.

```
models: [{
    model: [string] the name of the openerp model to load.
    label: [string] The label displayed during load.
    fields: [[string]|function] the list of fields to be loaded.
            Empty Array / Null loads all fields.
    order:  [[string]|function] the models will be ordered by the provided fields
    domain: [domain|function] the domain that determines what
            models need to be loaded. Null loads everything
    ids:    [[id]|function] the id list of the models that must
```

(continues on next page)

(продолжение с предыдущей страницы)

```
            be loaded. Overrides domain.
    context: [Dict|function] the openerp context for the model read
    condition: [function] do not load the models if it evaluates to
            false.
    loaded: [function(self,model)] this function is called once the
            models have been loaded, with the data as second argument
            if the function returns a deferred, the next model will
            wait until it resolves before loading.
 }]

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

В приведенном ниже примере все записи соответствуют модели домена “ account.invoice“.

** загруженная ** функция является обработчиком для загруженных данных.

Здесь вы можете продолжить и сохранить этот пример <https://github.com/it-projects-llc/pos-addons/blob/d0323907e35082d6d10416c2f7ef8497aa47dc31/pos_invoice_pay/static/src/js/main.js#L51-L64::>‘__ берется из модуля‘ *Оплата заказов и счетов через POS*“:

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

Данные пользовательского заказа в хранилище браузера

Перед тем как платежные поручения в POS хранятся в хранилище браузера. Таким образом, если мы снова откроем модуль POS (не следует путать с закрытием сессии), система автоматически извлекает данные из хранилища.

Если ваша модель добавляет данные (поля), вам необходимо выполнить дополнительную обработку данных, чтобы сохранить эти данные среди повторных открытий.

Поскольку хранилище браузера (* localStorage *) позволяет сохранять данные только с типом String, POS преобразует объект “ Order“ в String и обратно.

Для этого используются следующие методы:

- “ init_from_JSON“: функция считывает параметры порядка из * json-String * и сохраняет в текущий объект (см. реализацию Orderline здесь и для Модели‘ <https://github.com/odoo/odoo/blob/8d7ee3921384ce070d3333cbc4073ffc4f8febc4/addons/point_of_sale/static/src/js/models.js#L2024-L2082>‘__)

- “ export_as_JSON“: функция преобразует текущий объект в * json-String * (см. реализацию строки заказа здесь и для Модели‘ <https://github.com/odoo/odoo/blob/8d7ee3921384ce070d3333cbc4073ffc4f8febc4/addons/point_of_sale/static/src/js/models.js#L1558-L1576>‘__)

Когда заказ обновляется, вызывается “ export_as_JSON“ и данные сохраняются в хранилище браузера.

Теперь, если вы закрываете страницу и снова открываете ее, то в какой-то момент вызывается “ init_from_JSON“ для восстановления порядка из * строки json *.

Давайте возьмем пример:

Модуль “ POS Advanced Order Notes“

Этот модуль позволяет добавлять заметки ко всему заказу, использовать уже заданные заметки и ускорить процесс создания заказов, указав товары также с помощью заметок, которые могут автоматически применяться в дальнейшем.

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

Когда вы добавляете примечание к своему заказу, запускается триггер, который вызывает “ export_as_JSON“, чтобы преобразовать данные текущего заказа в строку (включая примечания) и сохранить ее в хранилище браузера.

При загрузке POS, чтобы получить сохраненные заметки из хранилища браузера, вам необходимо расширить функцию “ init_from_JSON“.

Без этого кода ваш модуль будет работать, но если вы перезагрузите POS, ваши заметки не будут представлены (потому что они не сохранены).

Отправка POS-заказов на сервер

В этой статье описывается процесс отправки POS-заказов на сервер odoo и демонстрируется возможное использование его расширения.

Общий процесс выглядит следующим образом:

Сторона клиента:

- “ export_as_JSON“: конвертирует * данные заказа * для отправки на сервер
- затем * заказ * сохраняется в кеш браузера
- затем POS пытается отправить данные на сервер

Бэкэнд сторона:

- “ create_from_ui“: данные поступают в POS (см. здесь <https://github.com/odoo/odoo/blob/33f1e5f64be0113e4e3ad7cb8de373d8ab5daa7b/addons/point_of_sale/models/pos_order.py#L722-L751>‘__)

- “ _process_order“: порядок обработки json (созданные записи в базе данных и т. д. см. здесь <https://github.com/odoo/odoo/blob/33f1e5f64be0113e4e3ad7cb8de373d8ab5daa7b/addons/point_of_sale/models/pos_order.py#L116-L155>‘__)
- “ _order_fields“: подготовить словарь для метода create (смотрите how)

Итак, чтобы передать дополнительную информацию и обработать ее на сервере, нам нужно:

- расширить “ export_as_JSON“ на стороне клиента
- расширить “ _process_order“ на стороне сервера

Давайте проверим это на примере:

Модуль “ Сохранение удаленных продуктов POS-заказа“

Модуль позволяет добавить причину отмены заказа или строки заказа в POS.

Для этого мы:

- расширить “ export_as_JSON“ на стороне клиента (см. здесь <https://github.com/it-projects-llc/pos-addons/blob/c5539c847d0656f6885087e27e497b8d985f1e31/pos_order_cancel/static/src/js/models.js#L138-L144>‘__)

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

- расширить “ _process_order“ на стороне сервера (см. здесь <https://github.com/it-projects-llc/pos-addons/blob/c5539c847d0656f6885087e27e497b8d985f1e31/pos_order_cancel/models/models.py#L56-L62>‘__)

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

2.10.5 Мгновенная синхронизация

POS Longpolling

Это пользовательский модуль odoo <https://github.com/it-projects-llc/pos-addons/tree/12.0/pos_longpolling>‘__" OOO "ИТ-Проекты", <https://it-projects.info>‘__, который позволяет отправлять мгновенные обновления на интерфейсы POS из серверной части.

Он предоставляет следующие методы в * Backend side *:

- “ self.env [’pos.config] .send_to_all_poses (channel_name, data) “: передает сообщения всем открытым POS (см. пример ‘ <https://github.com/it-projects-llc/pos-addons/blob/28d2b00bfd3f5d09bb65d5bf3245a6b87ed1d67b/pos_longpolling/models/pos_longpolling_models.py#L49-L53>‘__)
- “ pos_set._send_to_channel (channel_name, data) : передает сообщение POS в `` pos_set (см. пример ‘ <https://github.com/it-projects-llc/pos-addons/blob/28d2b00bfd3f5d09bb65d5bf3245a6b87ed1d67b/pos_longpolling/models/pos_longpolling_models.py#L22-L31>‘__)
- “ _send_to_channel_by_id (self, dbname, pos_id, channel_name) : отправляет сообщение для точного POS `` pos_id, использует имя базы данных “ dbname“, “ channel_name“, “ message = ’PONG’“ *(см. пример* <https://github.com/it-projects-llc/pos-addons/blob/28d2b00bfd3f5d09bb65d5bf3245a6b87ed1d67b/pos_longpolling/models/pos_longpolling_models.py#L34-L38>‘__)

Примечание: POS получит уведомление только в том случае, если он подписан на указанное “ имя_канала“.

Для * клиентской стороны * методы:

- “ add_bus (key, sync_server) : позволяет создать дополнительную шину для синхронизации данных с другим сервером синхронизации (см. пример ` <https://github.com/it-projects-llc/pos-addons/blob/4b9385b71f13f5df993317196d23972b65a7c2f8/pos_multi_session/static/src/js/pos_multi_session.js#L146>`__ в` `pos_multi_session - он получает данные с локального сервера для ускорения синхронизации)

Примечание: Вам не нужно использовать “ add_bus“, если вы подключаетесь к обычному серверу odoo.

- “ add_channel_callback: function (channel_name, callback, thisArg) “: подписка на определенный канал (см. пример ‘ <https://github.com/it-projects-llc/pos-addons/blob/28d2b00bfd3f5d09bb65d5bf3245a6b87ed1d67b/pos_longpolling/static/src/js/pos_longpolling.js#L97>‘__)

Давайте проверим пример использования, взяв за основу “ Sync Partners in POS“ модуль:

Синхронизация партнеров в POS-модуле

Этот ‘модуль <https://github.com/it-projects-llc/pos-addons/blob/907b16cc3a4ea613bf4fc81891a03739405e57a7/pos_partner_sync/> «__» при каждом обновлении партнера (в Backend) уведомляет POS об обновлении данных партнера.

Здесь вы можете увидеть, как он использует “ pos_longpolling“:

BACKEND

- При обновлении партнера метод “ send_field_updates“. <https://github.com/it-projects-llc/pos-addons/blob/907b16cc3a4ea613bf4fc81891a03739405e57a7/pos_partner_sync/models/res_partner.py#L39-L43::>‘__ называется:

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

- Он использует метод “ send_to_all_poses“.

КЛИЕНТ

- При запуске POS он подписывается на канал <https://github.com/it-projects-llc/pos-addons/blob/e471b4af2f062852d256d46c200e582b0f20d0ad/pos_partner_sync/static/src/js/pos_partner_sync.js#L13-L19::>“__“ *pos_partner_sync*“.

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

- Об уведомлении on_barcode_updates вызывается, который перезагружает данные партнера:

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

Поддержка нескольких сессий

“ pos_multi_session“ является модулем, который позволяет синхронизировать данные в POS в пределах одной multi_session.

Для синхронизации новых пользовательских данных Order или Orderline моделей одного POS с другими вам не нужно добавлять новый модуль “ pos_multi_session“ в “ depen“ вашего модуля, вам

нужно расширить такие методы, как “ export_as_JSON , `` init_from_JSON и добавьте метод “ apply_ms_data“, который используется для совместимости с.

Рассмотрим ** Пример синхронизации для модели заказа. **

Позвольте нам иметь некоторые данные для заказа, и нам нужно синхронизировать их со всеми POS-терминалами, которые используют один и тот же мульти-сеанс:

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

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

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

2.10.6 Продвинутая разработка POS

Дом Кеш

Dom Cache используется для сохранения визуализированных элементов для ускорения POS.

Чтобы добавить что-то в Dom Cache, вам нужно сделать что-то вроде этого:

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

Чтобы восстановить визуализированный элемент из кэша, сделайте что-то вроде этого

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

Вот полный пример из модуля Point of Sale <https://github.com/odoo/odoo/blob/12.0/addons/point_of_sale/static/src/js/screens.js#L761-L789>‘__:

Целью этого кода является оптимизация рендеринга элементов в POS. Каждая новая загрузка POS использует данные из ‘‘ DomCache‘‘ - тем самым экономит время на рендеринг новых элементов.

Давайте возьмем пример:

Модуль ‘‘ POS Order History‘‘

В этом в этом модуле <https://github.com/it-projects-llc/pos-addons/blob/12.0/pos_orders_history/static/src/js/screens.js#L159-L198>‘__‘ *DomCache‘* используется при отображении списка заказов.

После первой загрузки POS-элементы заказов, которые были обработаны (* HTML code *), сохраняются в Cache.

После перезагрузки POS проверяется наличие сохраненных элементов в Cache, и эти данные используются при обработке заказов.

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

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

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

(continues on next page)

(продолжение с предыдущей страницы)

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

JS доступ и наследование

action_button

Здесь вы найдете объяснение того, как получить / наследовать POS-объекты action_button.

Например, у нас есть определение в этом файле <https://github.com/odoo/odoo/blob/9.0/addons/pos_reprint/static/src/js/reprint.js#L1>‘_

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

Это определение не возвращает класс ReprintButton. Таким образом, мы не можем наследовать это обычным способом.

Чтобы достичь этого объекта, нам нужно получить его экземпляр, используя “ gui“. Тогда мы можем унаследовать это

Чтобы понять, на что это похоже, посмотрите пример, где гости отображают кнопки с номерами:

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

Хотя вы можете выполнять вызов и даже заменять функцию новой, вы не можете создавать наследование с помощью функций “ extend“ или “ include“. Это потому, что мы не можем добраться до класса и получить доступ только к экземпляру этого класса.

Такой подход имеет смысл только для тех виджетов:

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

screen_classes

Чтобы создать новый экранный виджет (с помощью метода extend ()) или изменить существующий экранный виджет (с помощью метода include ()), вам нужен целевой класс. Обычно вы можете получить этот класс, используя следующий код:

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

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

screens.OrderWidget.include({
    ...
```

Но это доступно только для виджетов, которые возвращаются основной функцией в файле "point_of_sale / static / src / js / screens.js".

** Список экранов: **

- ReceiptScreenWidget
- ActionButtonWidget
- define_action_button
- ScreenWidget
- PaymentScreenWidget
- OrderWidget
- NumpadWidget
- ProductScreenWidget
- ProductListWidget

В других случаях вы можете получить целевой класс экранного виджета, используя следующий код:

```
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({
    ...
```

Список экранов, доступных через “ screen_classes“:

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

2.11 Доступ

2.11.1 Руководство по безопасности

Ресурсы:

- 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

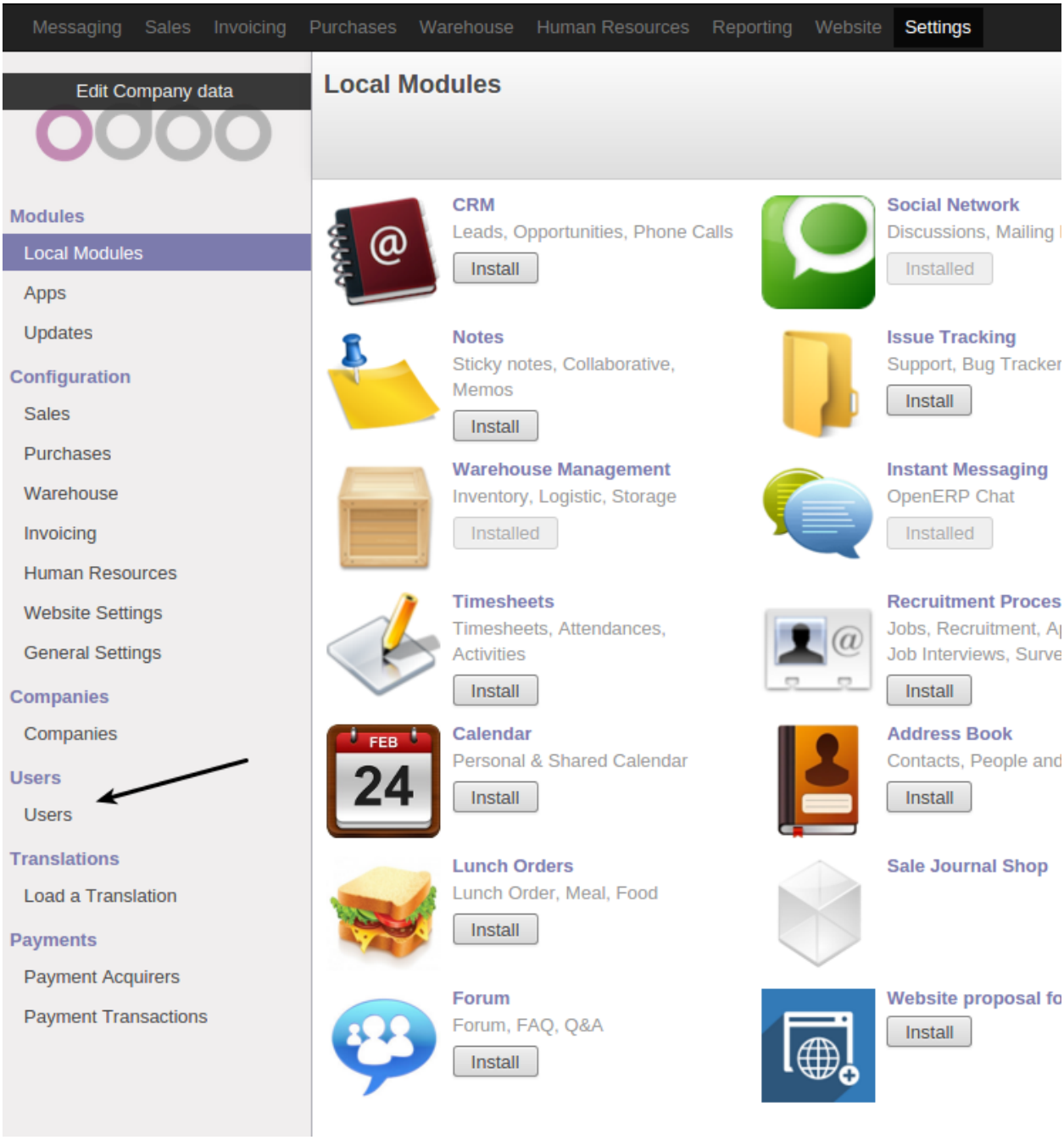
Odoo очень гибок в вопросах безопасности. Мы можем контролировать то, что пользователи могут делать, а что нет на разных уровнях. Также мы можем независимо контролировать каждую из четырех основных операций: чтение, запись, создание, отсоединение. Т.е. разрешить только чтение, разрешить только создание, предоставить разрешение на создание или удаление только.

На уровне полей / меню мы можем:

- скрыть поля или меню для одних пользователей и показать их для других
- сделать поля доступными для чтения только для некоторых пользователей и сделать их редактируемыми для других
- показать разные варианты выбора в полях выбора для разных пользователей

На полях уровня безопасности используются модели ‘‘ res.users‘‘ и ‘‘ res.groups‘‘. Эти модели относятся друг к другу так же, как и многие другие. Это означает, что пользователь может быть членом многих групп, и одна группа может быть назначена многим пользователям.

Одним из примеров того, как мы можем скрыть меню в отношении групп текущего пользователя, является следующее.



На картинке выше в “ Settings / Users“ мы видим только меню “ Users“. Мы знаем, что должно быть меню “ Groups“. Давайте посмотрим в `. / Openerp / addons / base / res / res_users_view.xml` о том, как пункт меню можно скрыть.

```
<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
```

(continues on next page)

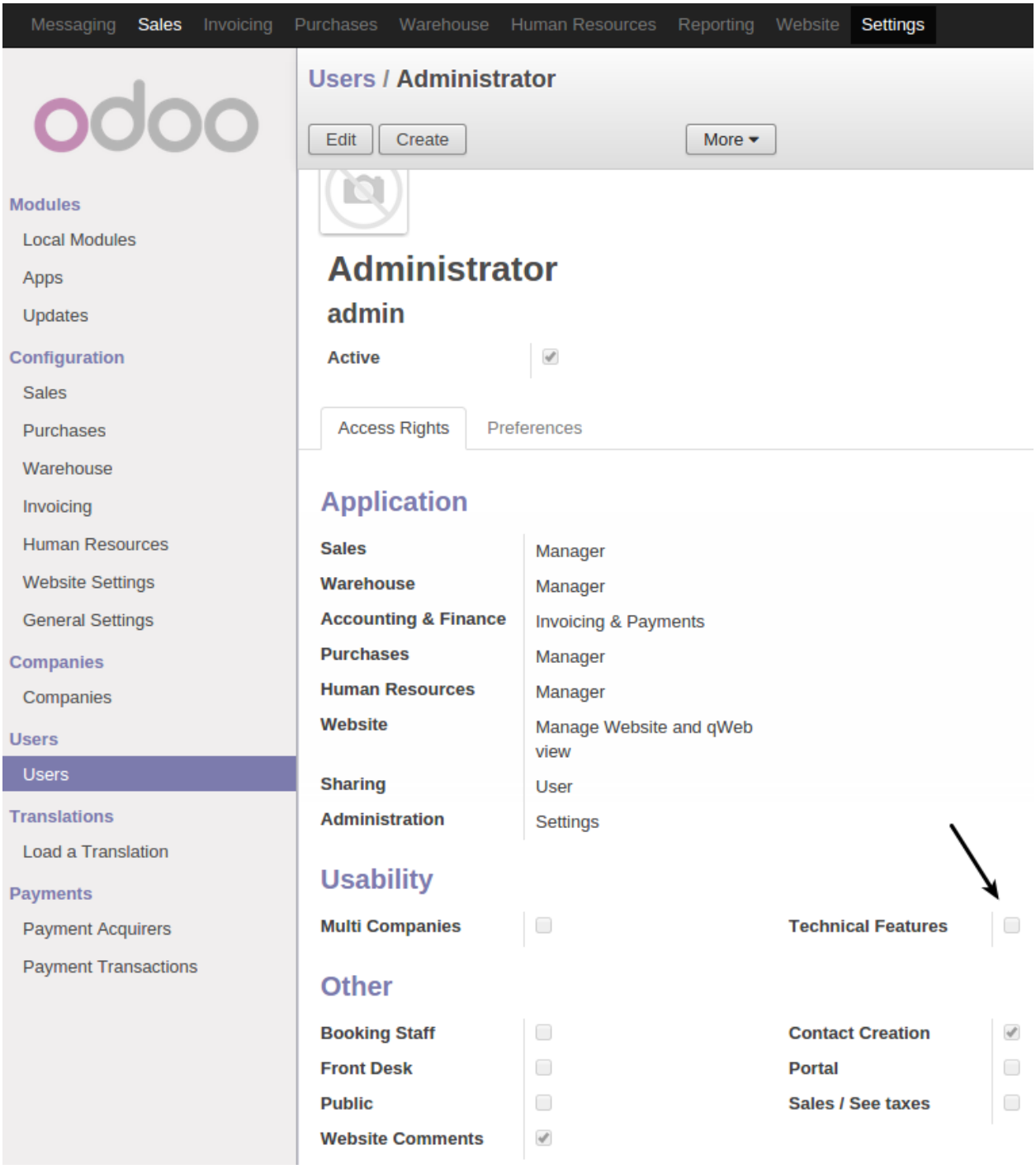
(продолжение с предыдущей страницы)

```
   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"/>
```

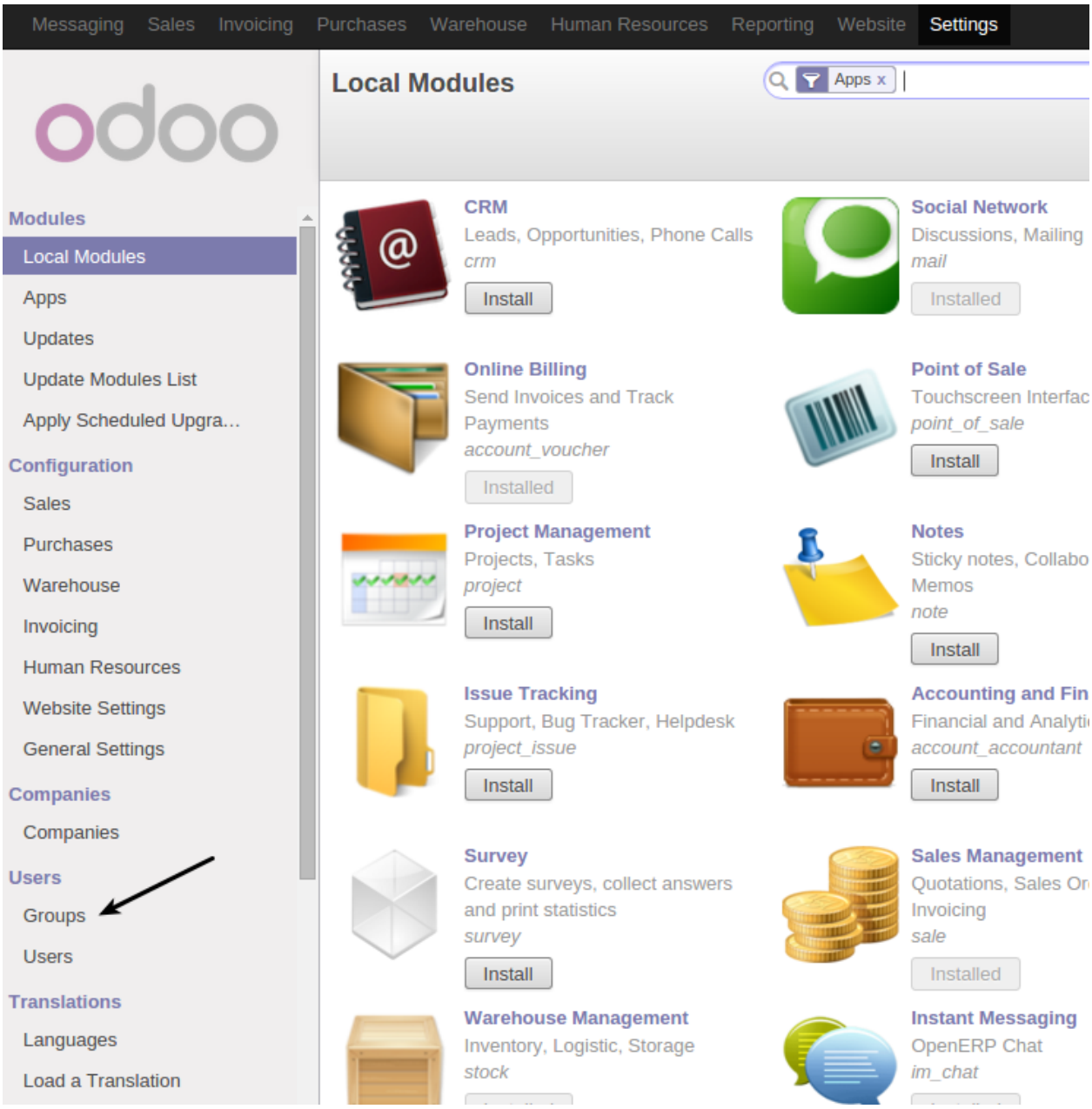
Атрибут “ groups“ в элементе “ menuitem“ показывает, что только члены группы “ base.group_no_one“ могут видеть пункт меню “ Groups“. Xmlid “ base.group_no_one“ ’определен в `. / Openerp / addons / base / security / base_security.xml` следующим образом.

```
<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>
```

Здесь мы можем увидеть “ group_no_one“ вместе с другими базовыми группами. Обратите внимание, что “ group_no_one“ имеет название “ Technical Features“. Давайте включим нашего пользователя в группу «Технические характеристики». Поскольку у нас нет доступа к пункту меню “ Groups“, единственный способ сделать это - это пункт меню “ Users“. Смотрите картинку ниже.



Установите флажок “ Технические характеристики“ и перезагрузите odoo. Теперь мы можем видеть пункт меню “ Groups“!



Из “ Настройки / Пользователи / Группы“ мы можем увидеть список существующих групп. Здесь мы также можем назначить пользователей для групп.

Скрыть поля

В `. / Openerp / addons / base / res / res_users_view.xml` мы можем видеть представление “ view_users_simple_form“. Обратите внимание, что поле “ company_id“ видимо только для членов группы “ base.group_multi_company“.

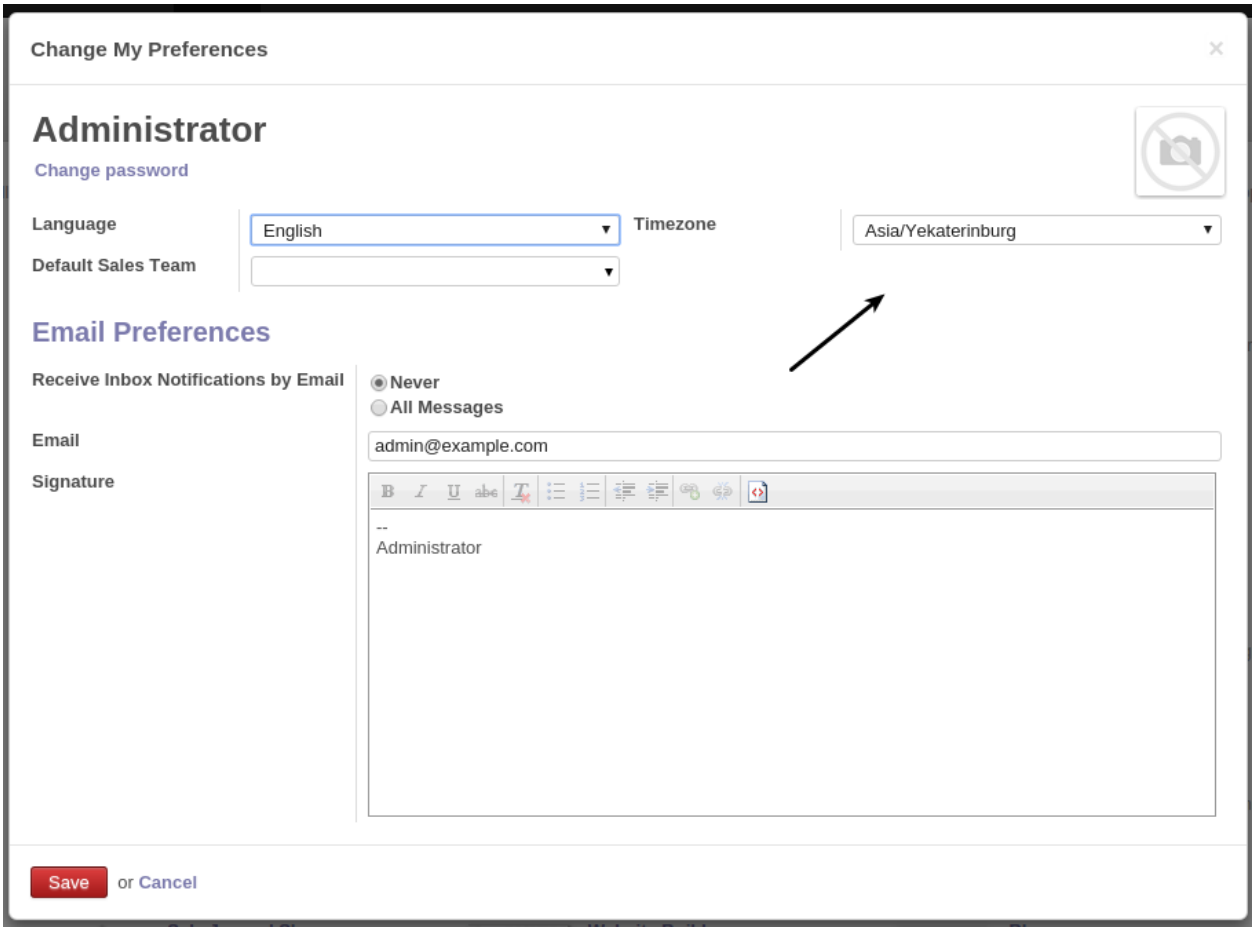
```
<!-- res.users -->
<record id="view_users_simple_form" model="ir.ui.view">
    <field name="name">res.users.simplified.form</field>
    <field name="model">res.users</field>
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    <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>
```

Наш текущий пользователь - Администратор. По умолчанию он не является членом группы “ base.group_multicompany“. Вот почему “ company_id“ не отображается для него в форме.



Модельные записи:

- ограничить доступ к указанному подмножеству записей в модели

Модель:

- ограничить доступ ко всем записям модели

2.11.2 Права суперпользователя

Администратор, то есть пользователь с идентификатором 1 (“ SUPERUSER_ID“), имеет исключения относительно прав доступа.

ir.model.access

Если какая-то модель не имеет записей в *ir.model.access (Access Rules)*, то только администратор имеет доступ к этой модели.

Смотрите также:

- *ir.model.access*
- *ir.rule*

2.11.3 Видео уроки

- Правила доступа (русский)

2.12 Крючки

2.12.1 post_load

- *Что мы знаем из комментариев в odoo source?*
- *Для чего это на самом деле?*
- *Пример патча обезьяны в odoo*
- *Почему мы должны использовать “ post_load“ для применения патча обезьяны?*
- *Как использовать post_load?*
- *Пример?*
- *Что-то еще нам нужно знать?*
- *Другое использование post_load?*

Что мы знаем из комментариев в 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.
```

Для чего это на самом деле?

Для ** Обезьяньих патчей **

Пример патча обезьяны в odoo

```
from odoo import tools

def new_image_resize_images(...)
    ...

tools.image_resize_images = new_image_resize_images
```

Почему мы должны использовать “ post_load“ для применения патча обезьяны?

Примечание: С ОДО 12 <https://github.com/odoo/odoo/commit/8226aa1db828d2a559c7ffaa31a27ef3e5ba4d0b>‘_ monkey patch может быть применен без post_load, но все же рекомендуется использовать его, чтобы быть уверенным.

Потому что иначе патч обезьяны будет применяться каждый раз, когда он будет доступен в пути аддонов. Это происходит потому, что odoo загружает файлы Python модуля, если в модуле есть статическая папка (независимо от того, установлен модуль или нет - см. Метод “ load_addons“ в http.py файл исходного кода odoo).

Как использовать post_load?

Вам необходимо определить функцию, доступную в файле “ __init __. Py“ модуля. Затем установите имя этой функции в качестве значения атрибута “ "post_load" “ в манифесте модуля.

Пример?

Конечно. Например, из модуля телеграммы.

В * __ openerp __. Py *

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

В * __ 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__
```

Что-то еще нам нужно знать?

Да.

Кроме того, если вам нужно применить monkey patch перед любой другой инициализацией, модуль необходимо добавить в параметр *server_wide_modules*.

Другое использование post_load?

В случае расширения модулей pos-box (например, “ hw_escpos“), вам, вероятно, нужно использовать post_load, потому что импорт hw_escpos из вашего модуля запускает специфичную для posbox инициализацию.

Пример из модуля hw_printer_network:

В * __ манифесте __. Py *

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

В * __ init __. Py *

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

В * controllers / hw_printer_network_controller.py *

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

...

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

2.13 Источник Дайвинг

Источник Дайвинг это способ найти ответы на ваши вопросы.

2.13.1 Исходные Дайвинг Случаи

Этот раздел содержит живые примеры исходного дайвинга.

Каждый случай содержит описание проблемы и возможные решения. Используйте проблемы в качестве упражнений, а решения в качестве руководства.

Дело: «Преобразованный метод»

контекст

При портировании модуля mail_move_message в файле‘ static / src / js / mail_move_message.js <https://github.com/yelizariev/mail-addons/blob/9.0/mail_move_message/static/src/js/mail_move_

message.js>‘_ есть метод‘ session.web.form.FormOpenPopup (this) <https://github.com/yelizariev/mail-addons/blob/9.0/mail_move_message/static/src/js/mail_move_message.js#L64>‘_.

проблема

В 9.0 не найден такой объект. Какой объект будет аналогом объекта? Что нужно сделать, чтобы найти этот объект?

Решение

Possible solution

Методические рекомендации

Используйте шаблон ниже для новых случаев

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

Context
=======

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

* LINK1
* LINK2

Problem
=======

What we need to do. E.g. port module to 9.0

* LINK1
* LINK2

Solution
========

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

2.13.2 Обзор: «Преобразованный метод»

Довольно часто при портировании модуля с 8.0 на 9.0 возникает ситуация, когда 8.0 является объектом, а 9.0 отсутствует. И не понятно - устарел и был удален или переименован. В очень сложных случаях объект может быть переименован и изменен почти до неузнаваемости.

Для поиска нужно сделать несколько шагов:

1. Представление по умолчанию, что такой объект существует, но он был переименован.
2. Смотри, что делает этот объект.

3. Поиск по названию методов, содержащих данный объект, за исключением общих слов (например, init, start, destroy ...).
4. Если результат не найден, ищите по уникальным ключевым словам, которые можно найти, приведя объект.
5. Если что-то не дало результатов, то, возможно, объект удаляется как устаревший.

Case

Possible solution

2.14 Odoo Translation Framework

2.14.1 How to overwrite built-in translation

You may need to change built-in translation via a module. You could be done via data xml files by calling `_set_ids` method. For example, for menus it could look as following:

```
<function
    model="ir.translation"
    name="_set_ids"
    eval="('ir.ui.menu,name', 'model', 'ru_RU', [ref('project.menu_main_pm')], 'Проекты Компании',
↪'Project')"/>
```

2.15 корпия

2.15.1 Установка

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

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

# install autoflake
sudo pip install --upgrade autoflake

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

2.15.2 Общие линты

```
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'
```

(continues on next page)

(продолжение с предыдущей страницы)

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

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

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

Исправление Python линтов в Odoo

Все версии

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

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

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

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

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

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


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

Odoo 10-

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

@ api.one -> @ api.multi

```
# Note. This solution doesn't work on methods that call super (e.g. write, create methods) or has␣
↪to return value
# Note. This solution doesn't handle properly methods with kwargs
find . -type f -name '*.py' | xargs perl -i -p0e 's/'\
'@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'
```

Исправление ошибок в JavaScript в odoo

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

Исправление первых линтов в odoo

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

Исправление xml линтов в odoo

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

2.16 Другой

2.16.1 Динамические записи

Хотя *XML* позволяет создавать только * статические * записи, существует способ динамического создания записи с помощью кода Python. Например, вам нужны динамические записи, чтобы добавить поддержку для корпоративных и общественных выпусков или добавить некоторые записи для каждой компании в базе данных и т. Д.

Есть несколько способов выполнить код при установке:

- СДЕЛАТЬ
- СДЕЛАТЬ
- СДЕЛАТЬ

Проблема с динамическими записями заключается в том, что odoo рассматривает такие записи как записи, которые были в XML-файлах, но теперь удалены. Это означает, что odoo удалит такие динамические записи сразу после обновления. Есть два способа решить это.

noupdate = False

Просто добавьте update = True в вашу запись “ ir.model.data“:

```
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

Если по какой-то причине вы не можете использовать noupdate = False, вы можете использовать следующий трюк.

Вот пример из модуля “ web_debranding“. Для создания записей в “ ir.model.data“ мы используем имя “ _web_debranding“. Тогда odoo будет считать такие записи принадлежащими другому модулю (“ _web_debranding“) и не удалит их. Но это также означает, что odoo не удалит их после удаления. Для дальнейшего случая нам нужно использовать “ uninstall_hook“.

Содержание

- *noupdate = False*
- *noupdate = True*
  - *файл питона*
  - *файл yaml*
  - *__openerp__.py*
  - *__init__.py*

файл питона

```
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,
        })
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        return view_id
```

файл yaml

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

__openerp__.py

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

__init__.py

```
from openerp import SUPERUSER_ID

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

2.16.2 База данных Odoo

Много ко многим

Для каждого * много ко многим * поля odoo создайте новую таблицу отношений, например * pos_multi_rel * с постфиксом * _rel *.

2.16.3 Оду способ шамана

** Что делать, если что-то не работает, но должен **

1. Обновить страницу
2. Модуль обновления
3. Проверьте файл openerp ** зависит **, ** демо ** и другие важные поля
4. Проверьте конфигурацию odoo, которую вы используете для запуска odoo. Особенности аддонов путей
5. Удалите и установите заново модули в зависимости
6. Очистить кеш браузера
7. Внимательно проверяйте логи. Посмотрите, если нужные файлы загружены или нет. Могут быть некоторые ошибки.

8. Создайте новую базу и установите все модули.

Глава 3

Отладка

В этом разделе описывается, как найти причину существующей проблемы.

3.1 Терминальные журналы

** Журналы ** из ** терминала ** * (в среде разработки) * или ** файла журнала ** * (в производственной среде) * являются основным источником для поиска причины проблемы.

Для управления уровнем вывода используйте - - *log-handler*

3.1.1 Выходной формат

Формат по умолчанию выглядит следующим образом:

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

```
2017-12-23 10:32:59,388 13  INFO  point_of_sale-10 werkzeug: 172.17.0.1 - - [23/Dec/2017 10:32:59]
↪"POST /web/webclient/translations HTTP/1.0" 200 -
asctime_____________ PID LEVEL DB_NAME_________ NAME____  MESSAGE_______________________________
↪_____________________________________________
```

имя

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

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

то есть равно названию пакета

PID

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

Сообщение

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

3.2 Консоль браузера

Консоль браузера (короткое имя: * console *) может содержать пользовательские журналы о клиентской части.

Чтобы открыть консоль Нажмите F12 в браузере.

3.3 Вкладка Источники в инструментах разработчика браузера

Позволяет проверить, какие файлы на стороне клиента загружены, а какие нет. Сделать это:

1. Включить * режим отладки (с активами) *
2. Откройте Инструменты разработчика (F12), перейдите на вкладку Источники и перезагрузите страницу.
3. Откройте левую панель (если она еще не открыта) и найдите интересующее приложение.

Пример: *Missing dependencies error in console*

3.4 Вкладка «Сеть» в инструментах разработчика браузера

Иногда ошибки не выводятся ни в Terminal. Затем вы можете попробовать найти полезную информацию на вкладке «Сеть» инструментов разработчика браузера.

3.4.1 Значение ответа

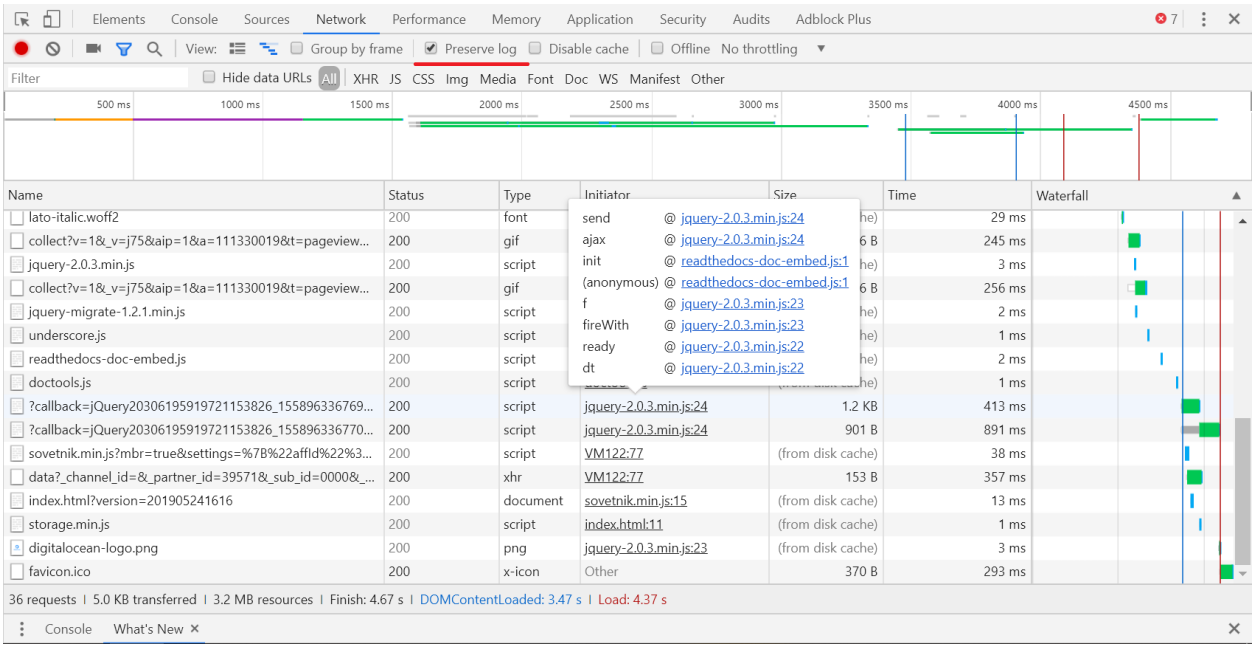
Чтобы увидеть ответ, щелкните строку запроса и перейдите на вкладку «Ответ».

3.4.2 Кто сделал http запрос

Предположим, мы хотим знать, какая часть нашего скрипта инициирует запрос. Для этого поместите указатель мыши над элементом столбца инициатора.

3.4.3 Сохранить журнал

Отметив флажок ** Сохранить журнал **, вы сохраните вывод консоли при обновлении страницы и закрытии / повторном открытии инструментов разработчика браузера. История консоли будет очищена только когда вкладка закрыта или вы вручную очистите консоль.



Примечание: Чтобы увидеть оригинальные файлы odoo js, т.е. не свернутые версии, сначала откройте odoo в * режиме отладки (с ресурсами) *

3.5 Qweb

Реализация javascript QWeb предоставляет несколько отладочных хуков:

“ **T-log“** принимает параметр выражения, вычисляет выражение во время рендеринга и записывает его результат с помощью “ console.log“

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

выведет “ 42“ на консоль

“ **T-debug“** запускает точку останова отладчика во время рендеринга шаблона:

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

остановит выполнение, если отладка активна (точное условие зависит от браузера и его инструментов разработки)

“ **T-js“** тело узла - это код JavaScript, выполняемый во время рендеринга шаблона. Принимает параметр “ context“, который является именем, под которым контекст рендеринга будет доступен в теле “ t-js“

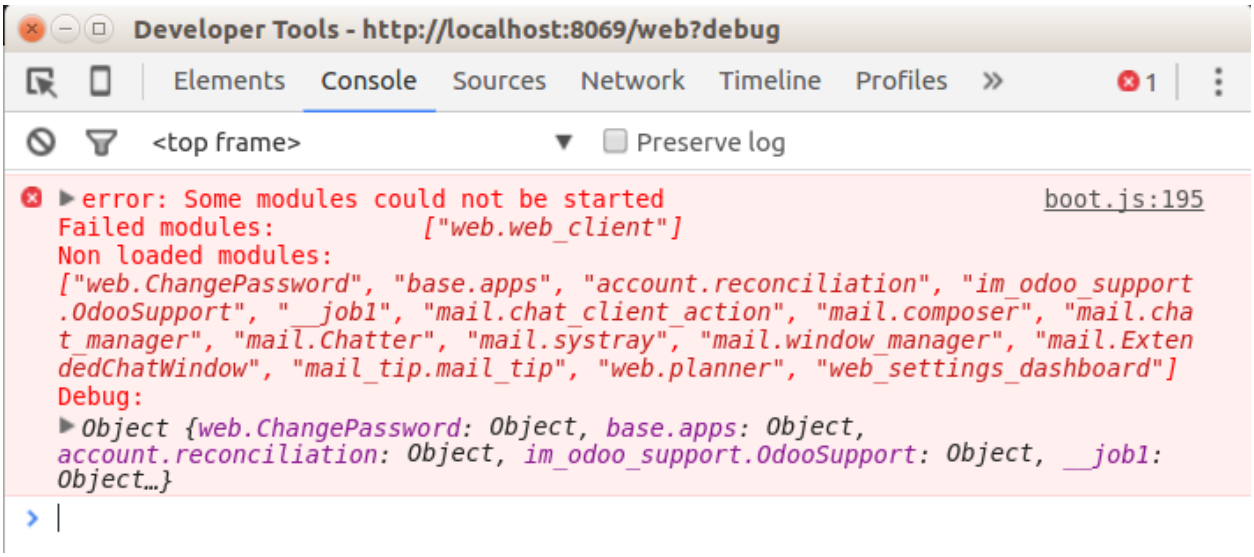
```
<t t-set="foo" t-value="42"/>
<t t-js="ctx">
    console.log("Foo is", ctx.foo);
</t>
```

Источник

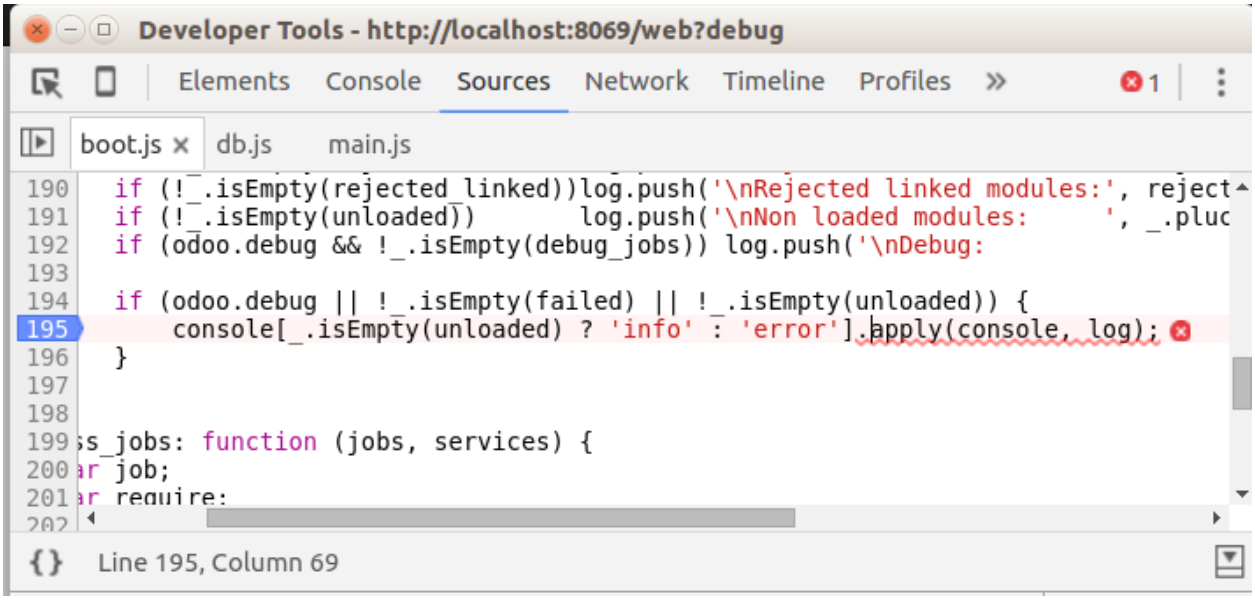
3.6 Типичные ошибки

3.6.1 Ошибка: сбойные модули

Если в консоли сервера нет ошибок, но boot.js выдает исключение, которое определяет причину ошибки, следующие шаги:



1. Перейти к строке ошибки в boot.js.
2. Включите точку останова.

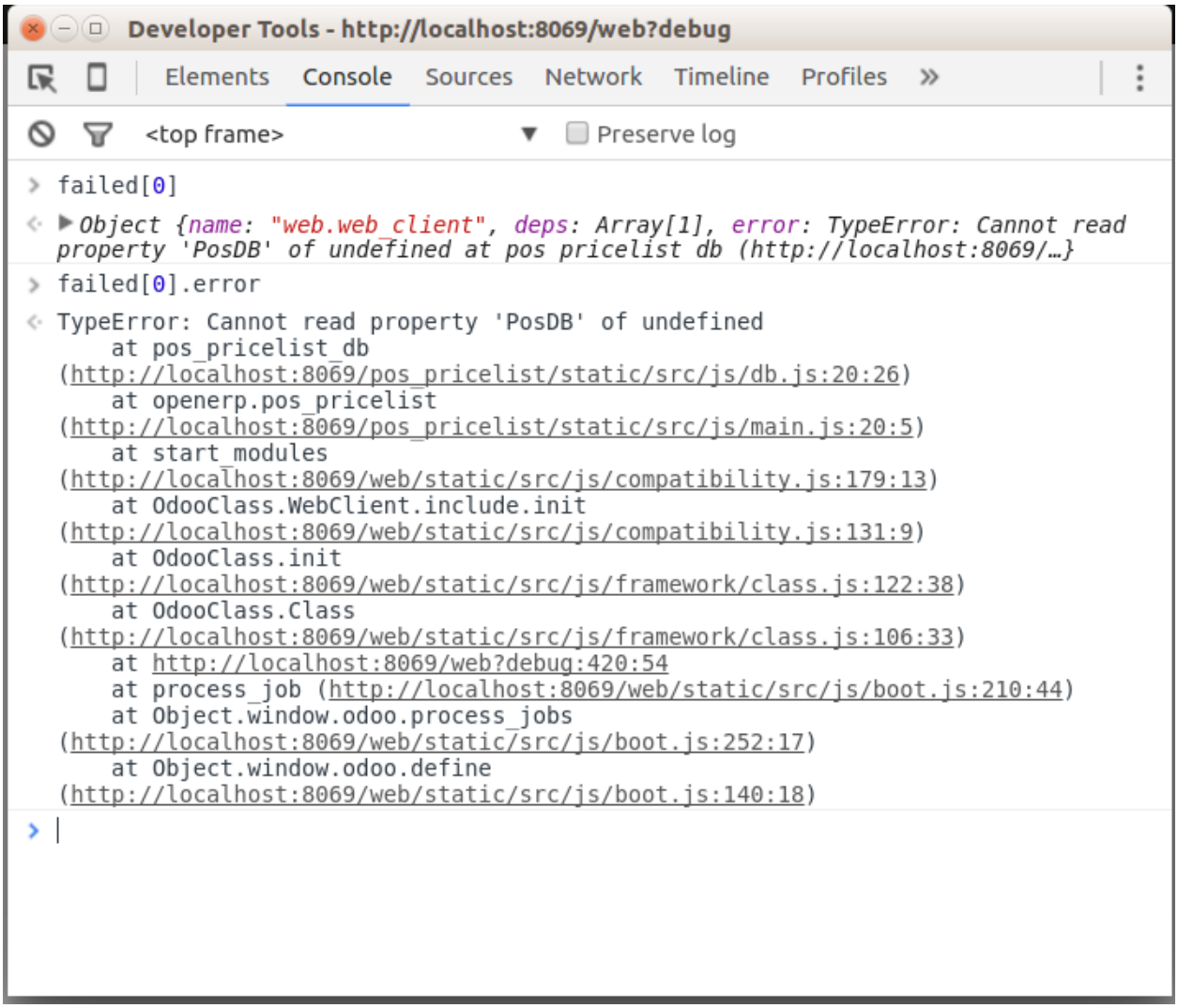


3. Сценарий повторного запуска (нажмите F5)
4. Когда скрипт остановится на строке ошибки, перейдите в консоль.

5. Введите команду

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

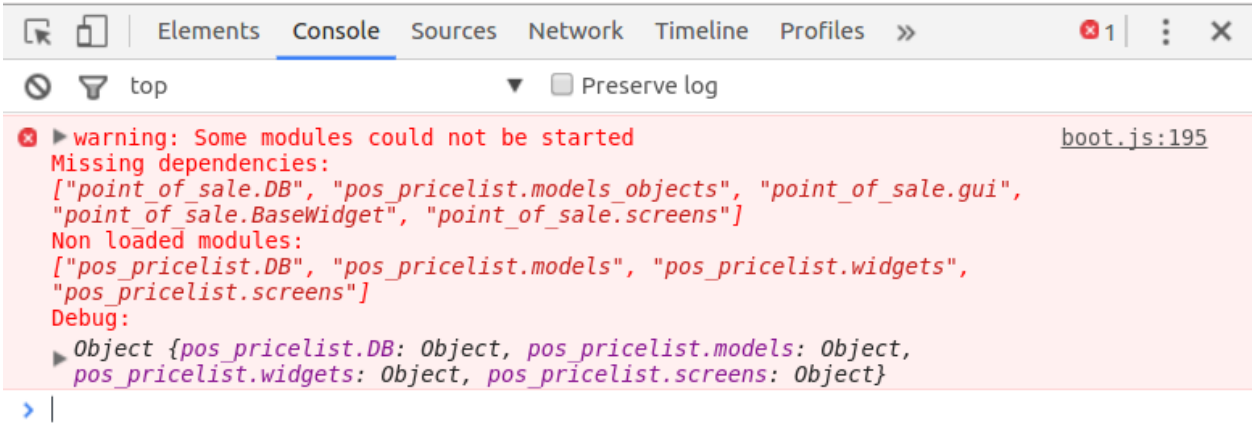
6. Чтобы получить вывод



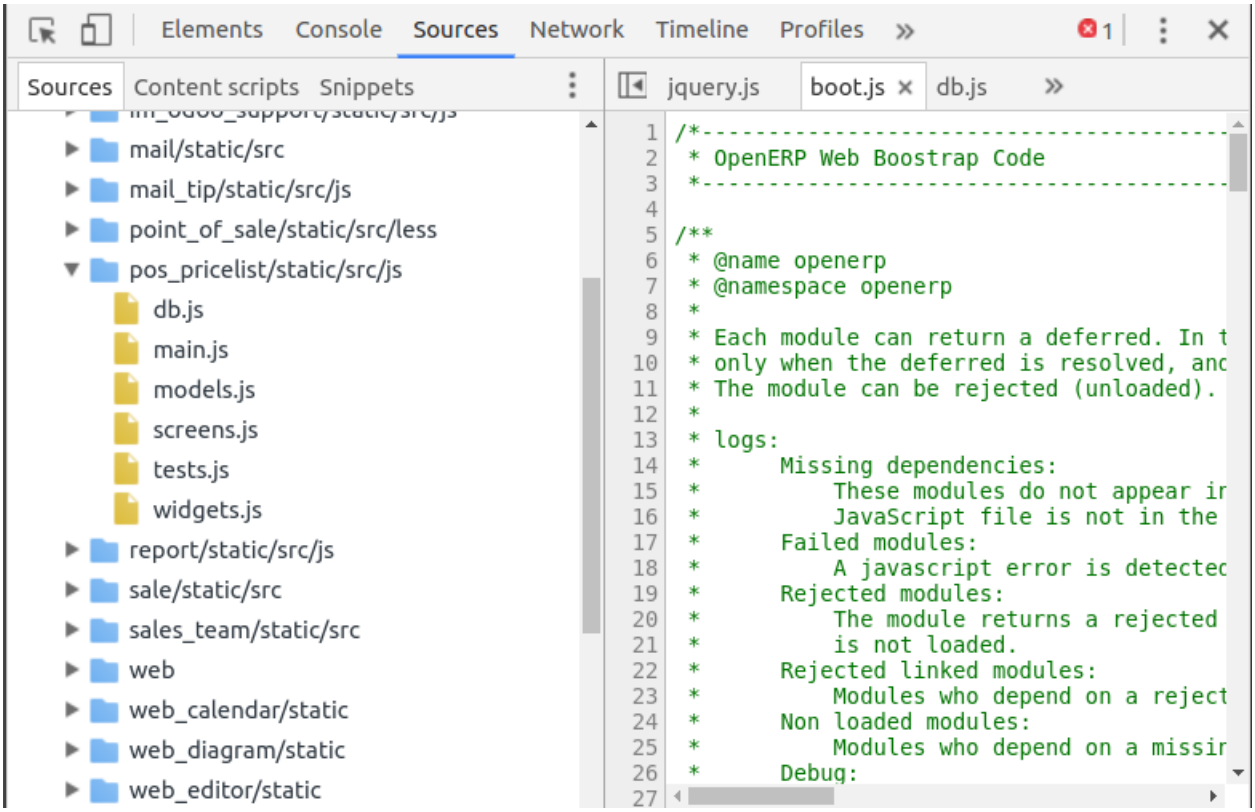
3.6.2 Ошибка: отсутствуют зависимости

Например, иногда при загрузке страницы отображается тип ошибки:

“ Отсутствуют зависимости: [. . .] Не загруженные модули: [. . .] “



Вы можете узнать причину в инструменте разработчика на вкладке Sources, как описано выше.

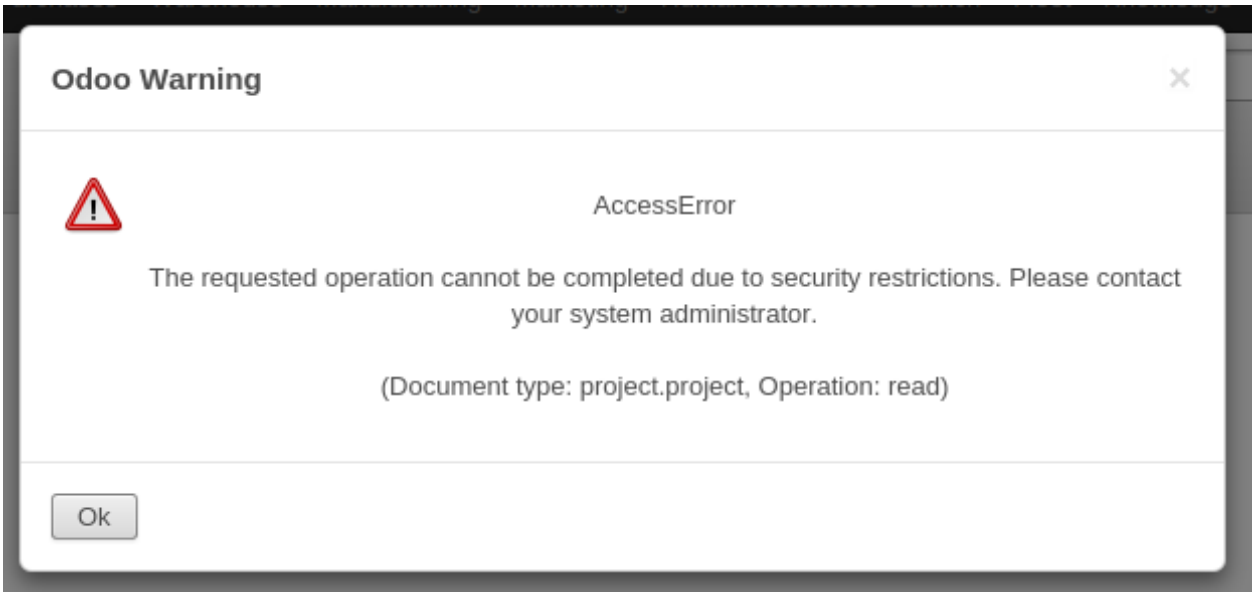


Скорее всего, вы не можете найти файлы, включенные в список отсутствующих зависимостей. Затем вам нужно проверить, включены ли они в файлы представлений (.xml).

3.6.3 AccessError: Пожалуйста, свяжитесь с вашим системным администратором

Существует ошибка AccessError, которая не определяет группы, которые имеют доступ к операции. Это просто заявляет:

> Запрошенная операция не может быть завершена из-за ограничений безопасности. Пожалуйста, обратитесь к системному администратору.



Такая ошибка означает, что ваш пользователь не удовлетворяет требованиям доступа, указанным в *ir.rule*. Смотрите *ir.rule* для общего понимания того, как работает защита odoo.

3.6.4 Исключение: автобус. Автобус недоступен

```
Traceback (most recent call last):
File "/odoo/odoo-server/odoo/http.py", line 650, in _handle_exception
  return super(JsonRequest, self)._handle_exception(exception)
File "/odoo/odoo-server/odoo/http.py", line 310, in _handle_exception
  raise pycompat.reraise(type(exception), exception, sys.exc_info()[2])
File "/odoo/odoo-server/odoo/tools/pycompat.py", line 87, in reraise
  raise value
File "/odoo/odoo-server/odoo/http.py", line 692, in dispatch
  result = self._call_function(**self.params)
File "/odoo/odoo-server/odoo/http.py", line 342, in _call_function
  return checked_call(self.db, *args, **kwargs)
File "/odoo/odoo-server/odoo/service/model.py", line 97, in wrapper
  return f(dbname, *args, **kwargs)
File "/odoo/odoo-server/odoo/http.py", line 335, in checked_call
  result = self.endpoint(*a, **kw)
File "/odoo/odoo-server/odoo/http.py", line 936, in __call__
  return self.method(*args, **kw)
File "/odoo/odoo-server/odoo/http.py", line 515, in response_wrap
  response = f(*args, **kw)
File "/odoo/odoo-server/addons/bus/controllers/main.py", line 37, in poll
  raise Exception("bus.Bus unavailable")
Exception: bus.Bus unavailable
```

Ошибка выше означает, что вы не настроили *longpolling* должным образом. Longpolling используется для мгновенных уведомлений и обновлений. Если вы уверены, что вам это не нужно, вы можете игнорировать ошибку.

Чтобы исправить ошибку, проверьте следующую страницу: *How to enable Longpolling in odoo*

3.6.5 ValueError: Внешний идентификатор не найден в системе: web.login

```
2019-05-28 08:51:28,012 13 INFO pr11 werkzeug: 172.17.0.1 - - [28/May/2019 08:51:28] "GET /web/
↪login HTTP/1.0" 500 -
2019-05-28 08:51:28,024 13 ERROR pr11 werkzeug: Error on request:
Traceback (most recent call last):
  File "/usr/local/lib/python3.5/dist-packages/werkzeug/serving.py", line 205, in run_wsgi
    execute(self.server.app)
  File "/usr/local/lib/python3.5/dist-packages/werkzeug/serving.py", line 193, in execute
    application_iter = app(environ, start_response)
  File "/mnt/odoo-source/odoo/service/wsgi_server.py", line 166, in application
    return application_unproxied(environ, start_response)
  File "/mnt/odoo-source/odoo/service/wsgi_server.py", line 154, in application_unproxied
    result = handler(environ, start_response)
  File "/mnt/odoo-source/odoo/http.py", line 1319, in __call__
    return self.dispatch(environ, start_response)
  File "/mnt/odoo-source/odoo/http.py", line 1293, in __call__
    return self.app(environ, start_wrapped)
  File "/usr/local/lib/python3.5/dist-packages/werkzeug/wsgi.py", line 599, in __call__
    return self.app(environ, start_response)
  File "/mnt/odoo-source/odoo/http.py", line 1491, in dispatch
    result = ir_http._dispatch()
  File "/mnt/odoo-source/odoo/addons/base/ir/ir_http.py", line 212, in _dispatch
    return cls._handle_exception(e)
  File "/mnt/odoo-source/odoo/addons/base/ir/ir_http.py", line 182, in _handle_exception
    return request._handle_exception(exception)
  File "/mnt/odoo-source/odoo/http.py", line 771, in _handle_exception
    return super(HttpRequest, self)._handle_exception(exception)
  File "/mnt/odoo-source/odoo/http.py", line 310, in _handle_exception
    raise pycompat.reraise(type(exception), exception, sys.exc_info()[2])
  File "/mnt/odoo-source/odoo/tools/pycompat.py", line 87, in reraise
    raise value
  File "/mnt/odoo-source/odoo/addons/base/ir/ir_http.py", line 208, in _dispatch
    result = request.dispatch()
  File "/mnt/odoo-source/odoo/http.py", line 830, in dispatch
    r = self._call_function(**self.params)
  File "/mnt/odoo-source/odoo/http.py", line 342, in _call_function
    return checked_call(self.db, *args, **kwargs)
  File "/mnt/odoo-source/odoo/service/model.py", line 97, in wrapper
    return f(dbname, *args, **kwargs)
  File "/mnt/odoo-source/odoo/http.py", line 338, in checked_call
    result.flatten()
  File "/mnt/odoo-source/odoo/http.py", line 1270, in flatten
    self.response.append(self.render())
  File "/mnt/odoo-source/odoo/http.py", line 1263, in render
    return env["ir.ui.view"].render_template(self.template, self.qcontext)
  File "/mnt/odoo-source/odoo/addons/base/ir/ir_ui_view.py", line 1211, in render_template
    return self.browse(self.get_view_id(template)).render(values, engine)
  File "/mnt/odoo-source/odoo/addons/base/ir/ir_ui_view.py", line 1118, in get_view_id
    return self.env['ir.model.data'].xmlid_to_res_id(template, raise_if_not_found=True)
  File "/mnt/odoo-source/odoo/addons/base/ir/ir_model.py", line 1358, in xmlid_to_res_id
    return self.xmlid_to_res_model_res_id(xmlid, raise_if_not_found)[1]
  File "/mnt/odoo-source/odoo/addons/base/ir/ir_model.py", line 1349, in xmlid_to_res_model_res_id
    return self.xmlid_lookup(xmlid)[1:3]
  File "<decorator-gen-21>", line 2, in xmlid_lookup

  File "/mnt/odoo-source/odoo/tools/cache.py", line 89, in lookup
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    value = d[key] = self.method(*args, **kwargs)
  File "/mnt/odoo-source/odoo/addons/base/ir/ir_model.py", line 1338, in xmlid_lookup
    raise ValueError('External ID not found in the system: %s' % xmlid)
ValueError: External ID not found in the system: web.login
```

Вышеуказанная ошибка обычно означает, что при инициализации базы данных возникла другая проблема. Итак, если вы получили такую ошибку в тестовой базе данных, просто удалите базу данных, начните создание базы данных снова и обратите внимание на журналы ошибок.

Если вы получили такую ошибку в производственной базе данных, это может быть трудно исправить. Извините ¯ \ _ () _ / ¯

Глава 4

Гарантия качества

Примечание: The section moved and now is available here.

Глава 5

Модули портирования

Предупреждение: this section is moved to https://itpp.dev/port/

Глава 6

Git и Github

6.1 Начальная конфигурация git & github

6.1.1 SSH ключи

Настройте github ssh-ключи: https://help.github.com/articles/connecting-to-github-with-ssh/

6.1.2 ключи gpg

- Создать ключи gpg: https://help.github.com/articles/generating-a-new-gpg-key/
- Добавьте ключ gpg в github: https://help.github.com/articles/adding-a-new-gpg-key-to-your-github-account/
- Скажите git, какой ключ использовать https://help.github.com/articles/telling-git-about-your-gpg-key/
- Скажите git подписать все коммиты:

```
git config --global commit.gpgsign true
```

- Заставьте gpg запомнить ваш пароль

```
# 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
```

(continues on next page)

(продолжение с предыдущей страницы)

```
# You may need to set GPG_TTY:
echo "export GPG_TTY=\"$( tty )\"" >> ~/.bashrc


# restart gpg-agent
gpgconf --kill gpg-agent
gpg-agent --daemon
```

- Сделайте резервную копию, если это необходимо

```
# 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
```

Предупреждение: Если вы потеряли свой ключ или забыли пароль, вам нужно создать новый, но не удаляйте старый из github, потому что в противном случае все подписанные коммитом старого ключа станут «Непроверенными»

6.1.3 мерзкая электронная почта

- Настройка электронной почты в git. Email должен быть таким же, как в настройках github:

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

6.1.4 редактор git

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

6.1.5 gitignore

- Настройка глобального gitignore

  Возможное содержимое для “ ~ / .gitignore_global“:

```
*~
*.pyc
```

6.2 Портирование

Если вы добавили какую-то функцию в одну ветку и вам нужно добавить ее в другую ветку, то вам нужно сделать * port *.

Смотрите также:

- *Conflicts resolving*

6.2.1 Форвард-порт

Это самый простой случай. Вы объединяете коммиты из более старой ветви (например, 8.0) в более новую ветку (например, 9.0)

```
git checkout 9.0
git merge origin/8.0

# [Resolve conflicts if needed]

git push
```

После “ git merge“ вам, вероятно, понадобится внести небольшие изменения. В этом случае просто добавьте новые коммиты в более новую ветку

```
git add ...
git commit -m "...."
git push
```

6.2.2 Back-порт

Если вам нужно перенести новую функцию из более новой ветви (например, 9.0) в более старую (например, 8.0), то вы должны сделать * back-port *.

Проблема здесь в том, что в более новой ветви есть коммиты, которые должны применяться только для более новой ветви. То есть вы не можете просто сделать “ git merge 9.0“, потому что он приносит коммиты только 9.0 в ветку 8.0. Возможные решения здесь:

6.2.3 мерзавец

Применить коммиты из более новой ветви (например, 9.0) к более старой ветви (например, 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
```

Также можно выбрать коммит из любого удаленного репозитория. Добавьте этот репозиторий к вашим пультам. Извлекай из этого. А потом вишня.

Вишневый выбор коммитов

Команда “ git cherry-pick A..B“ применяет коммиты между A и B, но без A (A должен быть старше B). Чтобы применить диапазон коммитов, используйте формат следующим образом:

```
git cherry-pick A^..B
```

Например, чтобы зарегистрировать этот PR https://github.com/it-projects-llc/odoo-saas-tools/pull/286/commits, используйте команду

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

6.3 Разрешение конфликтов

После выполнения “ git merge“ или “ git cherry-pick“ могут возникнуть конфликты, потому что некоторые коммиты пытаются внести изменения в одну и ту же строку. Итак, вам нужно выбрать, какое изменение будет использоваться. Это может быть один вариант, оба варианта или новый вариант.

Что делать, если у вас возникли конфликты:

- Проверь состояние

  ```
  git status
  ```

- Разрешить конфликты:
  - либо отредактируйте файлы вручную:
    - открыть файл с конфликтами
    - найдите “ <<< “ или “ >>> “ и удалите устаревший вариант или сделайте смесь обоих вариантов.
  - или используйте следующие команды, если вы уверены, какая версия должна быть сохранена:

    ```
    git checkout --ours -- <file>
    # or
    git checkout --theirs -- <file>
    ```

- Пометить файлы как разрешенные с помощью команды “ git add“
- Выполнено.

  ```
  git push
  ```

6.3.1 Удаленные файлы

Иногда изменения могут быть противоречивыми, поскольку файлы больше не существуют в * нашей * версии, но обновляются в * их * (или наоборот). В этом случае выполните код ниже, чтобы игнорировать такие изменения:

```
git status | grep 'deleted by us' | awk '{print $4}' | xargs git rm
git status | grep 'deleted by them' | awk '{print $4}' | xargs git rm
```

6.3.2 Примечания

- Важно, чтобы на этапе разрешения конфликта вы не делали никаких обновлений внутри конфликтующих строк. Вы можете только выбрать, какие строки следует сохранить, а какие уда-

лить. Например, если вы разрешаете конфликты из-за переноса некоторого обновления функции из одной версии odoo (например, 8.0) в другую (например, 9.0), то такие изменения должны быть настроены некоторое время, чтобы заставить update feature работать с целевой версией odoo. Но вы должны сделать такую настройку только на новом коммите. Сделайте коммиты слияния chery-picking только о слияниях и chery-picking, сделайте коммиты портирования отдельно.

- Если у вас нет конфликтов, вам не нужно делать коммит после cherry-pick, потому что он создает коммит самостоятельно.

6.4 Multi Pull Request

6.4.1 Найти последнюю объединенную точку

Чтобы найти последний коммит “ upstream / 8.0“ и “ upstream / 9.0“, используйте следующие команды

```
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
```

(continues on next page)

(продолжение с предыдущей страницы)

```
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
```

6.5 Отмена хромого коммита

Представьте, что вы делаете хромой коммит. Сейчас для ремонта дела делаем следующее:

1. git reset HEAD ~ 1 –soft
2. статус мерзавца

Вы увидите: ваша ветвь отстает от 'origin / 8.0' на 1 коммит и может быть быстро перенесена. (используйте «git pull» для обновления вашей локальной ветки)

3. git add // Добавить сюда измененные (исправленные) файлы
4. git diff –cached // убедитесь, что все в порядке.
5. статус мерзавца

Вы увидите: ваша ветвь отстает от 'origin / 8.0' на 1 коммит и может быть быстро перенесена. (используйте «git pull» для обновления вашей локальной ветки)

6. git commit -m'I исправил свои ошибки '
7. статус мерзавца

Вы увидите: ваша ветка и 'origin / 8.0' разошлись и имеют 1 и 1 разные коммиты соответственно. (используйте "git pull", чтобы объединить удаленную ветку с вашей)

Теперь, наконец, сила с вами

8. git push origin 8.0 -f

6.6 Вытащить запрос из консоли

Да, это возможно! Попробуйте это руководство: https://github.com/github/hub Чем в консоли

```
alias git=hub
```

И тянуть запрос

```
git pull-request upstream 9.0
```

Необходимо добавить заголовок для запроса на получение. Сохрани это. Если все в порядке, вы получите ссылку на ваш запрос.

6.7 Проверьте удаленные пакеты

Проверьте текущую ветку

```
git branch -vv
```

Местная ветка должна быть привязана к источнику. Если его нет, делайте следующее:

```
git push -u origin 9.0-pos_ms
```

6.8 Перемещение файлов

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

6.8.1 git format-patch

Этот раздел основан на инструкции OCA. <https://github.com/OCA/maintainer-tools/wiki/Migration-to-version-10.0>‘_

Используемые переменные:

- “ $ REPO_PATH“, “ $ REPO_NAME“ - исходный репозиторий
- “ $ MODULE“ - название модуля, который вы хотите переместить
- “ $ BRANCH“ - ветвь $ REPO с $ MODULE
- “ $ DEST_REPO_PATH“, “ $ DEST_REPO_NAME“ - целевой репозиторий

```
# 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
```

6.8.2 git filter-branch

Этот раздел основан на http://gbayer.com/development/moving-files-from-one-git-repository-to-another-preserving-history

Цель:

- Переместите каталог 1 из Git-репозитория A в Git-репозиторий B.

Ограничения:

- Git-репозиторий A содержит другие каталоги, которые мы не хотим перемещать.
- Мы хотели бы сохранить историю коммитов Git для каталога, который мы перемещаем.

Давайте начнем

- $ REPO: репозиторий, содержащий модуль (например, “ misc-addons“)
- $ DEST_REPO: репозиторий, в который вы хотите переместить модуль (например, “ access-addons“)
- $ MODULE: имя модуля, который вы хотите переместить (например, “ group_menu_no_access“)
- $ BRANCH: ветвь $ REPO с $ MODULE (исходная ветвь, например, “ 8.0“)

Предупреждение: Если вы установили git из официального Deb-репозитория Ubuntu 14.04, то вам следует сначала обновить его. Вы можете обновить git, используя эту инструкцию *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
```

После последней команды у вас будет модуль со всеми его коммитами в вашем репо назначения. Теперь вы можете нажать его на github и т. Д. Вы можете удалить папку “ ~ / $ REPO“ - теперь она не используется.

Предупреждение: Клонирование - это обязательный шаг. Это временный каталог. Будут удалены все модули, кроме того, который вы хотите переместить.

Следующий скрипт может пригодиться, если вам нужно переместить несколько модулей. Но убедитесь, что вы понимаете все его команды перед использованием.

```
#!/bin/bash

source_repo=$PWD
echo $source_repo
```

(continues on next page)

(продолжение с предыдущей страницы)

```
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
```

Чтобы использовать его, вы должны сделать файл movemodule.sh в вашем домашнем каталоге и поместить все строки выше и сделать этот файл исполняемым.

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

Чтобы переместить group_menu_no_access из addons-yelizariev в access-addons с помощью movemodule.sh, выполните следующие действия.

```
$ cd ~
$ git clone https://github.com/yelizariev/addons-yelizariev.git
$ cd addons-yelizariev
```

Эта часть аналогична перемещению без сценария. Но тогда я набираю только одну команду вместо

многих в случае полностью ручного подхода.

```
addons-yelizarie$ ~/movemodule.sh group_menu_no_access ~/access-addons 8.0
```

6.9 Git тайник

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

6.10 Обновление Git

Официальный deb-репозиторий Ubuntu 14.04 имеет версию Git 1.9. Это слишком старый и должен быть обновлен.

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
```

6.11 Сквош фиксируется в одном

6.11.1 Резервный

Перед тем, как сделать сквош, подумайте о «резервном копировании» ваших коммитов.

Локальное резервное копирование:

```
git tag 9.0-new-module-backup
```

Удаленное резервное копирование

```
git push origin 9.0-new-module:9.0-new-module-backup
```

Для восстановления исходного состояния вы можете использовать следующую команду:

```
# 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
```

6.11.2 “ git commit –amend“

Вместо создания нового коммита, добавляет обновления к последнему коммиту.

6.11.3 “ git rebase -i“

Интерактивная сквош

```
git rebase -i <your-first-commit>^
# e.g.
git rebase -i 7801c8b^
```

Затем отредактируйте открытый файл и оставьте “ pick“ для первого коммита и замените “ pick“ на “ squash“ для остальных. Например

Происхождение

```
TODO
```

Отредактированный

```
TODO
```

Предупреждение: Если вы удалите строку здесь, то этот коммит будет потерян.

6.11.4 От себя

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

6.12 Создать ветку из чужого Pull Request

```
git fetch upstream pull/354/head:pr354
git checkout -b 10.0-branch-name pr354
```

Дополнительная информация: https://help.github.com/articles/checking-out-pull-requests-locally/

6.12.1 Push-обновления для чьего-либо запроса на извлечение

Если у вас есть доступ к редактированию PR-файлов через пользовательский интерфейс github, вы можете отправить такие обновления из консоли.

```
GITHUB_USERNAME=yelizariev  # set username where PR is made from
REPO=pos-addons # set repo name
BRANCH=10.0-fix-something # set source branch name

git remote add ${GITHUB_USERNAME} git@github.com:${GITHUB_USERNAME}/${REPO}.git
git fetch ${GITHUB_USERNAME} ${BRANCH}
git checkout ${GITHUB_USERNAME}/${BRANCH}
# make updates
# ...
# make commit
git commit ...

# push update to another's branch
git push ${GITHUB_USERNAME} HEAD:${BRANCH}
```

Глава 7

Непрерывная интеграция

7.1 Runbot

7.1.1 runbot.odoo.com

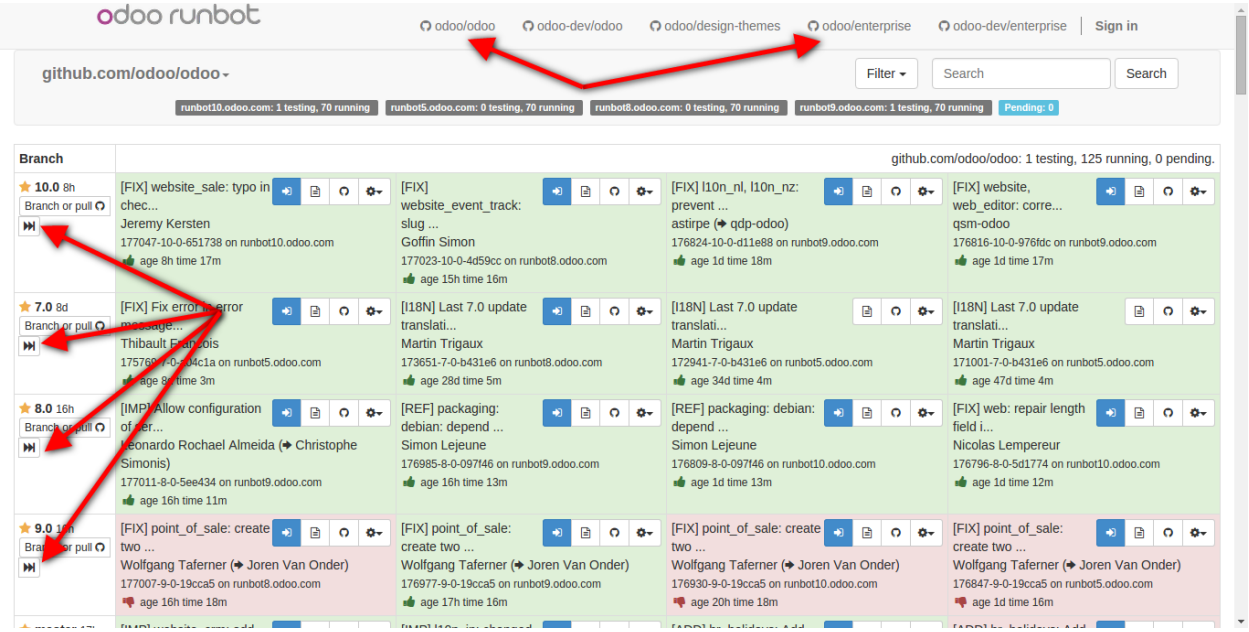
http://runbot.odoo.com/ - официальный runbot. Хотя его основная цель - проверка запросов на получение доступа к официальному репозиторию, он полезен в повседневной рутине разработки.

- Позволяет играть с любой версией odoo. В каждой сборке установлены все модули с демонстрационными данными.
- Это позволяет быстро попробовать корпоративные версии odoo

Как использовать runbot.odoo.com?

- открыть http://runbot.odoo.com/runbot/
- переключитесь на нужный вам репозиторий. Сообщество Odoo (odoo / odoo) по умолчанию.
- найдите строку с нужной вам версией odoo (10.0, 9.0, 8.0, 7.0)
- нажмите на значок * fast forward *, чтобы открыть последнюю сборку. Кроме того, нажмите любую синюю кнопку в строке, которая соответствует нужной версии odoo.
- на странице входа введите учетные данные:

- Администратор
  - Логин: “ admin“
  - пароль: “ admin“
- демонстрация
  - Логин: “ demo“
  - пароль: “ demo“



7.1.2 runbot.it-projects.info

http://runbot.it-projects.info/ - настраиваемый runbot для репозиториев IT-проектов.

Этапы изготовления сборки:

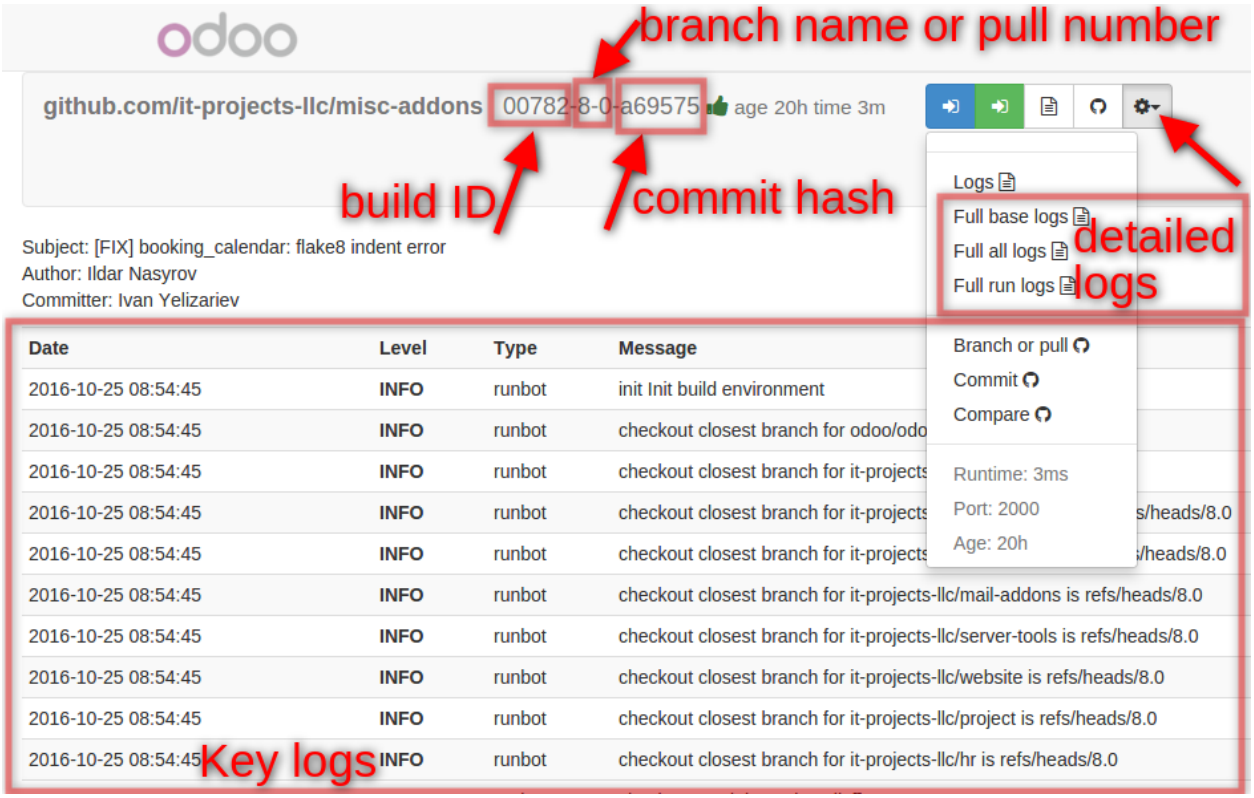
- Извлечь источники из github
- ** - база ** базы данных: установить обновленные модули для сборок по запросу и базовые модули для сборок веток. Для некоторых репозиториев также установлены явные модули (т.е. те, которые указаны в настройках runbot)
- ** - все ** базы данных: установить все модули репо
- запустить сборку с двумя подготовленными базами данных

Основные характеристики:

- Синяя кнопка - войти в ** - все ** базы данных
- Зеленая кнопка - войти в ** - базу ** базы данных
- Ключевые журналы (показанные на странице сборки) - ключевые журналы, предупреждения и ошибки
- Подробные журналы (текстовые файлы)
  - *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



7.1.3 Как развернуть runbot?

Существует докер, который позволяет вам развернуть свой собственный runbot для ваших репозиториев. Проверьте это для получения дополнительной информации

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

7.2 Одо Трэвис Тесты

СДЕЛАТЬ

7.3 покрытие

Глава 8

Odoo

8.1 модели

Раздел помогает в понимании встроенных моделей

8.1.1 ir.config_parameter

Добавить запись по модулю

XML: <record>

Код:

```
<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:

- запись удаляется при удалении

Минусы:

- выдает ошибку, если запись с этим ключом уже создана вручную

XML: <function>

Код:

```
<function model="ir.config_parameter" name="set_param" eval="('auth_signup.allow_uninvited', True,␣
↪['base.group_system'])" />
```

Prons:

- ошибка не возникает, если запись с этим ключом уже создана вручную

Минусы:

- запись не удаляется при удалении
- значение перезаписывается после каждого обновления модуля

YML

Примечание: Файлы Yaml не поддерживаются <https://odoo-development.readthedocs.io/en/latest/odoo/models/ir.config_parameter.html>‘__ odoo 12

Код:

```
-
  !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:

- значение не перезаписывается, если оно уже существует

Минусы:

- запись не удаляется при удалении

8.1.2 res.users

СДЕЛАТЬ

8.1.3 res.groups

СДЕЛАТЬ

8.1.4 ir.model.access

Определяет доступ ко всей модели.

Каждый элемент управления доступом имеет модель, которой он предоставляет разрешения, разрешения, которые он предоставляет, и, необязательно, группу.

Контроль доступа является аддитивным, для данной модели пользователь имеет доступ ко всем разрешениям, предоставленным любой из его групп: если пользователь принадлежит к одной группе, которая разрешает запись, и другой, которая позволяет удалять, они могут как писать, так и удалять.

Если группа не указана, контроль доступа применяется ко всем пользователям, в противном случае он применяется только к членам данной группы.

Доступные разрешения: создание (“ perm_create“), поиск и чтение (“ perm_read“), обновление существующих записей (“ perm_write“) и удаление существующих записей (“ perm_unlink“)

When there is no access records for a given model and permission (e.g. *read*), then only Superuser has the permision.

Смотрите также:

- *Superuser rights*
- : Документ: ‘ir.rule <ir.rule> ‘

поля

```
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')
```

8.1.5 ir.rule

Правила записи - это условия, которым записи должны удовлетворять для выполнения операции (создание, чтение, запись или удаление). Пример условия: * Пользователь может обновить задание, которое ему назначено *.

Поле группы определяет, для какого правила группы применяется. Если группа не указана, то правило является * глобальным * и применяется ко всем пользователям.

Доменное поле определяет условия для записей.

Булевы поля (чтение, запись, создание, удаление) ir.rule означают * Применить это правило для операций такого типа . *Они* * не ** означают * ограничивают доступ для такого рода операций *.

Проверка алгоритма доступа

Чтобы проверить, есть ли у пользователя доступ, например, к * чтению * записи, система делает следующее:

- Проверьте доступ в соответствии с: doc: ‘ir.model.access <ir.model.access> записи. Если это не проходит, то пользователь ** не получает ** доступ
- Найти и проверить общие правила для ** модели ** и для операции * read *
  - если запись ** не удовлетворяет ** (не подходит для домена) хотя бы по одному из глобальных правил, то пользователь ** не получает ** доступа
- Find and check non-global rules for the **model**, which *perm_ read* equal True and `groups` that intersect with current user groups.

- if there are no such rules, then user **get** access
- если запись ** удовлетворяет ** (соответствует домену) для ** хотя бы одного ** неглобальных правил, то пользователь ** получает ** доступ
- если запись ** не удовлетворяет ** для ** всех ** неглобальных правил, то пользователь ** не получает ** доступа

Смотрите также:

- *Superuser rights*

поля

```
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)
```

8.1.6 product.template

В магазинах есть товары, которые отличаются от других только одним или несколькими свойствами. Такие товары не имеет смысла отделять как отдельные товары. Они объединяются в группу аналогичных товаров, которые называются ** шаблон **.

** магазин: ** на страницах товара используется product.template (при создании заказа используется *product.product*).

8.1.7 product.product

Продукт, в отличие от шаблона *template*, это отдельный продукт, который можно рассчитать, установить цену, назначить скидку.

product.product используется:

- sale.order
- склад
- позиция

8.1.8 ir.actions.todo

Модель используется для выполнения действий (записи в модели "ir.actions.act_window"). Модель позволяет задавать условия и последовательность появления мастеров. Также вы можете указать обычное окно интерфейса, но только в качестве последнего действия. Код:

```
<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>
```

Тип запуска может быть одним из следующих:

- ручной: запускается вручную.
- автоматический: запускается всякий раз, когда система переконфигурируется. Запуск происходит либо после установки / обновления любого модуля, либо после вызова метода execute в модели res.config.
- один раз: после запуска вручную автоматически устанавливается значение «Готово».

8.1.9 bus.bus

автобус

Bus - это модуль для мгновенных уведомлений через longpolling. Добавьте его в список зависимостей:

```
'depends': ['bus']
```

Примечание: Почтовый модуль в odoo 9.0 уже зависит от шины модуля.

Предупреждение: Не путайте longpolling bus с *core.bus*, который только на стороне клиента и является частью модуля “ web“.

Что такое длинный опрос

- *About longpolling*
- *How to enable Longpolling in odoo*

Как реализовать лонгполлинг

- *Схема работы*
- *Идентификатор канала*
- *Прослушанные каналы*
- *Обязательное уведомление о событии*
- *Начать опрос*
- *Отправка уведомления*
- *Обработка уведомлений*

Схема работы

- Укажите каналы, которые слушает текущий клиент
- Привязать событие уведомления к вашему обработчику
- Начать опрос
- Отправить уведомление на некоторый канал через код Python

Идентификатор канала

Идентификатор канала - это способ отличить один канал от другого. В основном канал содержит dbname, некоторую строку и некоторый идентификатор.

Добавленные через js идентификаторы могут быть только строковыми.

```
var channel = JSON.stringify([dbname, 'model.name', uid]);
```

С помощью идентификаторов Python может быть добавлена строка или любая структура данных.

```
# tuple
channel = (request.db, 'model.name', request.uid)
# or a string
channel = '["%s","%s","%s"]' % (request.db, 'model.name', request.uid)
```

Предупреждение: JSON.stringify в js и json.dumps в python могут дать другой результат.

Прослушанные каналы

Вы можете добавить каналы двумя способами: либо на стороне сервера через функцию “ _poll“ в контроллере шины, либо в файле js, используя метод “ bus.add_channel () “.

С контроллерами:

```
# 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)
```

В файле js:

```
// 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);
```

Обязательное уведомление о событии

В файле js:

```
bus.on("notification", this, this.on_notification);
```

Начать опрос

В файле js:

```
bus.start_polling();
```

Примечание: Вам не нужно вызывать “ bus.start_polling (); “, если он уже был запущен другим модулем.

Когда начинается опрос, отправляется запрос “ / longpolling / poll“, так что вы можете найти и проверить его с помощью инструмента «Сеть» в своем браузере.

Отправка уведомления

Вы можете отправить уведомление только через питона. Если вам нужно сделать это через клиента, сначала отправьте сигнал на сервер обычным способом (например, через контроллеры).

```
self.env['bus.bus'].sendmany([(channel1, message1), (channel2, message2), ...])
# or
self.env['bus.bus'].sendone(channel, message)
```

Обработка уведомлений

```
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
        // ...
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    }
},
```

Примеры

** pos_multi_session: **

- добавить канал (питон)
- связать событие
- отправить уведомление

** шахматы: **

- добавить канал (js)
- связать событие
- отправить уведомление

** mail_move_message: **

- добавить канал (питон)
- связать событие
- отправить уведомление

8.1.10 ir.cron

** Создание автоматизированных действий в Odoo **

Планировщики - это автоматизированные действия, которые выполняются автоматически в течение определенного периода времени и могут выполнять множество задач. Они дают возможность выполнять действия базы данных без необходимости ручного взаимодействия. Odoo облегчает запуск фонового задания: просто вставьте запись в таблицу “ ir.cron“ и Odoo выполнит ее в соответствии с определением.

** 1. Создание модели и метод этой модели. **

```
class model_name(models.Model):
    _name = "model.name"
    # fields
    def method_name(self, cr, uid, context=None): # method of this model
        # your code
```

** 2. Создание автоматизированного действия **

> ‘Если вы хотите создавать новые модули в руководствах от Odoo, вы должны добавить код для автоматизированного действия в yourDefaultModule / data / в отдельный файл XML.

При использовании автоматических действий важно отметить, что они всегда должны быть определены в поле noupdate, поскольку их не следует обновлять при обновлении модуля.

```
<openerp>
    <data noupdate="1">
        <record id="unique_name" model="ir.cron">
```

(continues on next page)

(продолжение с предыдущей страницы)

```
            <field name="name">Name </field>
            <field name="active" eval="True" />
            <field name="user_id" ref="base.user_root" />
            <field name="interval_number">1</field>
            <field name="interval_type">days</field>
            <field name="numbercall">-1</field>
            <field name="doall">1</field>
            <!--<field name="nextcall" >2016-12-31 23:59:59</field>-->
            <field name="model" eval="'model.name '" />
            <field name="function" eval="'method_name '" />
            <field name="args" eval="" />
            <!--<field name="priority" eval="5" />-->
        </record>
    </data>
</openerp>
```

Первое, что вы заметите, это data “ noupdate = "1" “, это говорит Odoo, что весь код в этом теге не должен обновляться при обновлении вашего модуля.

```
<record id="unique_name" model="ir.cron">
```

Идентификатор является уникальным идентификатором для Odoo, чтобы знать, какая запись связана с каким идентификатором. Модель под названием ("ir.cron") - это модель, специально созданная Odoo для всех автоматизированных действий. Эта модель содержит все автоматизированные действия и всегда должна быть указана.

```
<field name="name">Name </field>
```

Следующая строка - это имя.

```
<field name="active" eval="True" />
```

Логическое значение, указывающее, активно ли задание cron.

```
<field name="user_id" ref="base.user_root"/>
```

Этот идентификатор пользователя относится к конкретному пользователю, в большинстве случаев это будет base.user_root.

```
<field name="interval_number">1</field>
```

Количество раз, которое планировщик должен быть вызван на основе "interval_type"

```
<field name="interval_type">days</field>
```

Интервальный блок.

В списке должно быть одно значение: “ minutes“, “ hours“, “ days“, “ days“, “ months“.

```
<field name="numbercall">-1</field>
```

Целочисленное значение, указывающее, сколько раз задание выполняется. Отрицательное значение означает отсутствие ограничений.

```
<field name="doall">1</field>
```

Логическое значение, указывающее, следует ли выполнять пропущенные вхождения при перезапуске сервера.

```
<field name="nextcall" >2016-12-31 23:59:59</field> <!-- notice the date/time format -->
```

Следующая запланированная дата выполнения для этой работы.

```
<field name="model" eval="'model.name '" />
```

Поле “ модель“ указывает, по какой модели должно вызываться автоматическое действие.

```
<field name="function" eval="'method_name '" />
```

Имя метода, вызываемого при обработке этого задания.

```
<field name="args" eval="" />
```

Аргументы для передачи в метод.

```
<field name="priority" eval="5" />
```

Приоритет задания, как целое число: 0 означает более высокий приоритет, 10 означает более низкий приоритет.

** Значения по умолчанию. **

имя	Определение
nextcall	“ лямбда * a: time.strftime (DEFAULT_SERVER_DATETIME_FORMAT“
приоритет	5
ID пользователя	“ лямбда-объект, cr, uid, context: uid“
interval_number	1
interval_type	месяцы
numbercall	1
активный	1
сделай все	1

8.1.11 mail.message

** Подтипы сообщений в Odoo **

Большую часть времени в Odoo несколько пользователей работают над одной конкретной записью или документом, таким как заказ на продажу, счет-фактура, задачи и т. Д. В таких сценариях становится чрезвычайно важно отслеживать изменения, внесенные каждым лицом в этот документ. Это помогает руководству найти любую возможную причину в случае возникновения любой проблемы. Odoo предоставляет эту функцию в значительной степени с помощью интеграции OpenChatter.

Рассмотрим сценарий, в котором несколько пользователей работают в одном проекте. Различные параметры для этого проекта уже настроены, например, крайний срок, изначально запланированные часы и т. Д. Теперь один из пользователей изменяет значение запланированных часов. Поэтому теперь важно знать, какой пользователь изменил его и какое значение было предыдущим. Мы можем отследить это, создав подтипы сообщений в Odoo следующим образом.

Это должно быть определено в XML, который будет иметь следующий синтаксис.

```
<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>
```

Пользователи также могут иметь тип mail.message.subtype, который зависит от другого, чтобы действовать через поле отношения. Для запланированных часов у нас может быть следующий синтаксис для него.

```
<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 предоставляет возможность отслеживать различные события, связанные с одним конкретным документом, с помощью атрибута _track. Если мы наследуем объект mail.thread, то с помощью атрибута _track пользователь может также отправлять уведомления, чтобы информировать других об изменениях, происходящих в этом конкретном документе. Синтаксис может быть следующим.

```
_track = {
    'planned_hours': {
    'project.mt_task_planned_hours': lambda self, cr, uid, obj, ctx=None: obj.planned_hours,
},
```

Чтобы отслеживать изменения, связанные с любым полем, Odoo предоставляет атрибут с именем track_visibility. Он должен быть определен на уровне поля, который имеет синтаксис ниже.

```
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.')
```

Следовательно, легко отслеживать изменения, сделанные до сих пор в отношении любого конкретного документа различными пользователями.

8.2 Как использовать Odoo

8.2.1 Как создать базу данных

Из пользовательского интерфейса

Для создания новой базы данных откройте “ / web / database / manager“

8.0-

База данных с точками

Ранняя версия odoo не позволяет создавать базы данных с точками. Вы можете снять это ограничение двумя способами:

1. Источники обновлений

```
cd path/to/odoo
sed -i 's/matches="[^"]*"//g' addons/web/static/src/xml/base.xml
```

2. обновить HTML-код с помощью инструмента * Inspect Element *

   Вы должны удалить значение поля совпадений.

От терминала

9.0+

Для создания новой базы данных просто добавьте параметр “ -d“ при запуске odoo, например:

```
./openerp-server -d database1
```

– will create new database with name `database1`

8.2.2 Как включить технические характеристики

8,0

- открыть “ Настройки / Пользователи / Пользователи“
- выберите своего пользователя
- нажмите “ [Изменить] “
- Включить “ Технические характеристики“
- нажмите “ [Сохранить] “
- обновить веб-страницу (нажмите “ F5“)

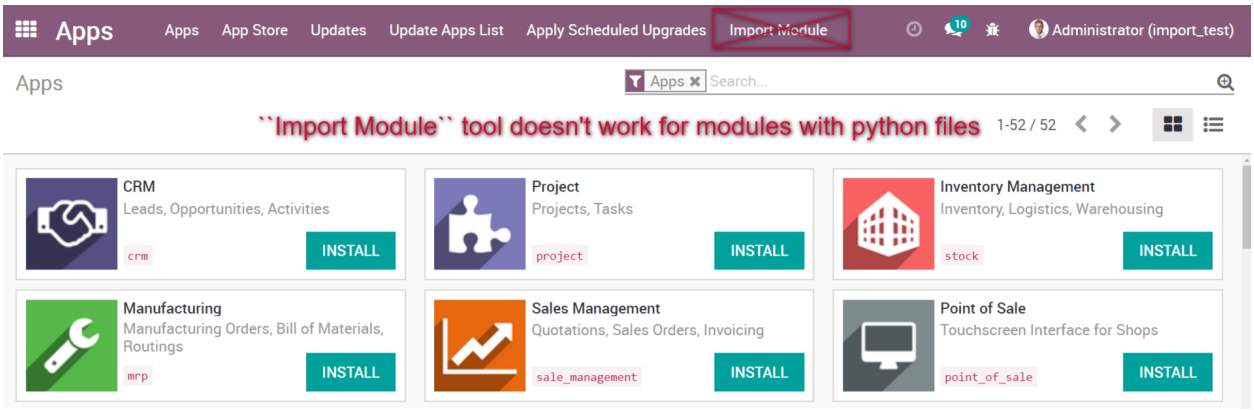
9.0+

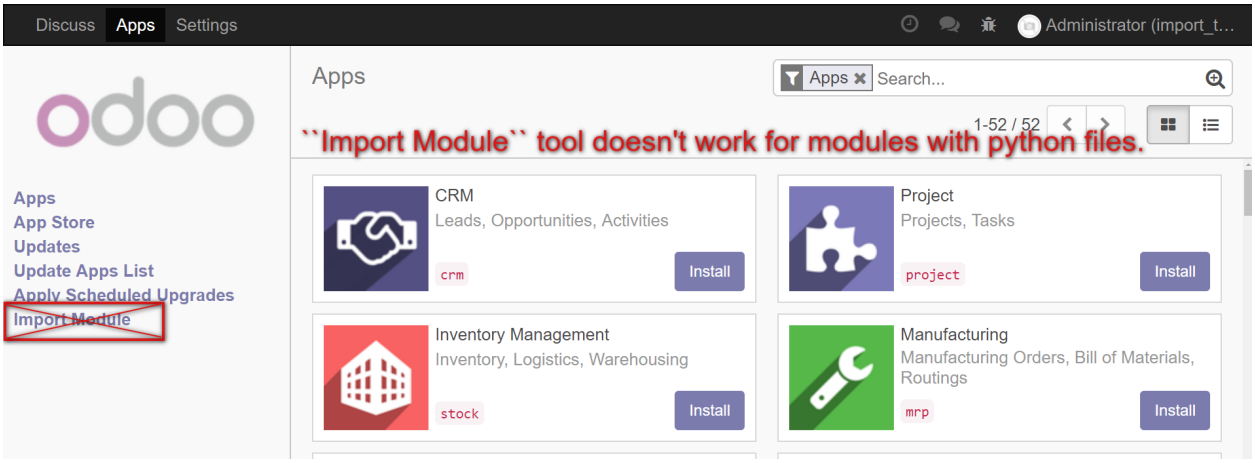
Начиная с Odoo 9.0 для включения технических функций, вам нужно только *activate developer mode*.

8.2.3 Как установить / обновить модуль

- *Из почтового архива*
  - *11.0+*
    - *устанавливать*
    - *Обновить*
  - *10.0+*
    - *устанавливать*
    - *Обновить*
  - *9,0*
    - *устанавливать*
    - *Обновить*
  - *8,0*
    - *устанавливать*
    - *Обновить*

Предупреждение: Инструмент “ Module Module“ (импорт из zip-файла) не работает для модулей с файлами Python. Это означает, что это не работает в большинстве случаев





Из почтового архива

- разархивируйте модуль в папку дополнений
- перезапустите сервер odoo

11.0+

устанавливать

- *activate developer mode*
- перейдите в меню “ Apps“
- нажмите «Обновить список приложений»
- найдите и откройте нужный вам модуль
- нажмите “ [Установить] “

Обновить

- перейдите в меню “ Apps“
- найдите и откройте нужный вам модуль
- нажмите “ [Обновить] “

10.0+

устанавливать

- *activate developer mode*
- перейдите в меню “ Apps“
- нажмите «Обновить список приложений»
- найдите и откройте нужный вам модуль

- нажмите “ [Установить] “

Обновить

- перейдите в меню “ Apps“
- найдите и откройте нужный вам модуль
- нажмите “ [Обновить] “

9,0

устанавливать

- *activate developer mode*
- перейдите в меню “ Apps“
- нажмите «Обновить список приложений»
- найдите и откройте нужный вам модуль
- нажмите “ [Установить] “

Обновить

- перейдите в меню “ Apps“
- найдите и откройте нужный вам модуль
- нажмите “ [Обновить] “

8,0

устанавливать

- перейдите к “ [[Настройки]] >> Локальные модули“
- найдите и откройте нужный вам модуль
- нажмите “ [Установить] “

Обновить

- перейдите к “ [[Настройки]] >> Локальные модули“
- найдите и откройте нужный вам модуль
- нажмите “ [Обновить] “

8.2.4 Как активировать режим разработчика

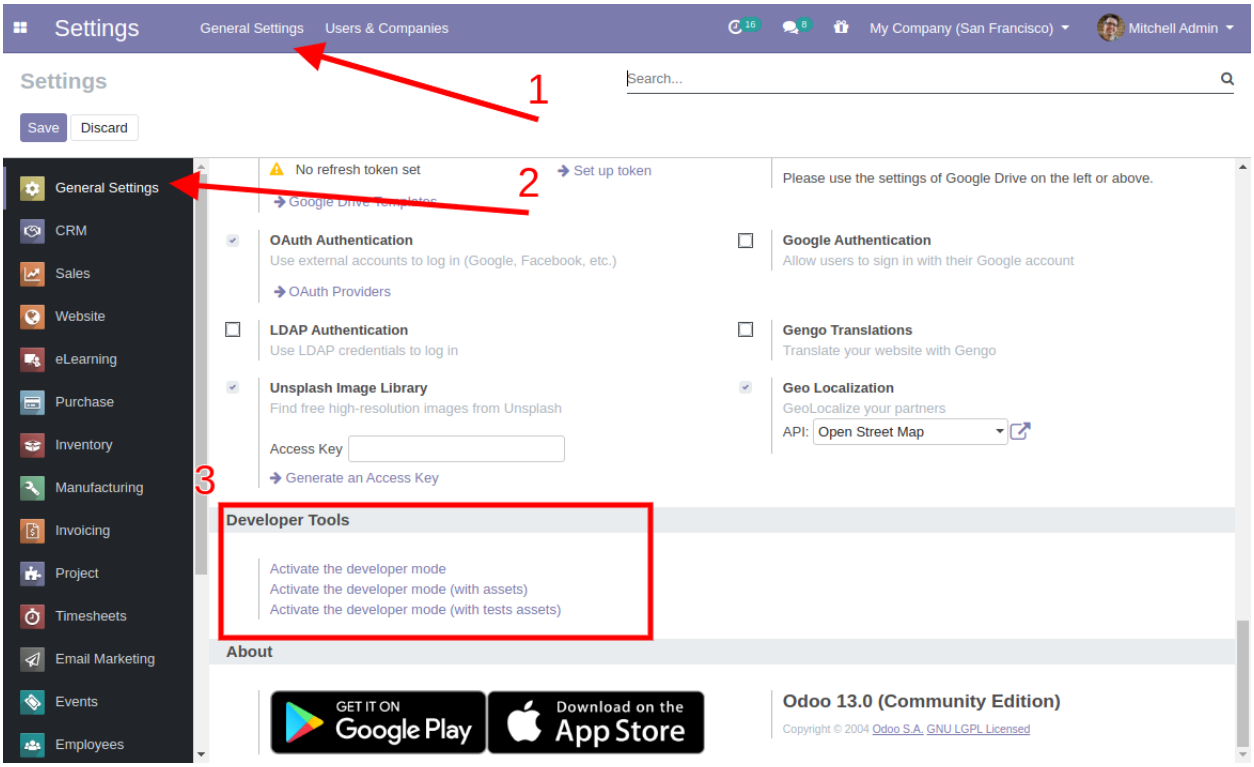
Добавьте параметр “ debug“ в ваш URL, например:

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

или используйте пользовательский интерфейс, как описано ниже

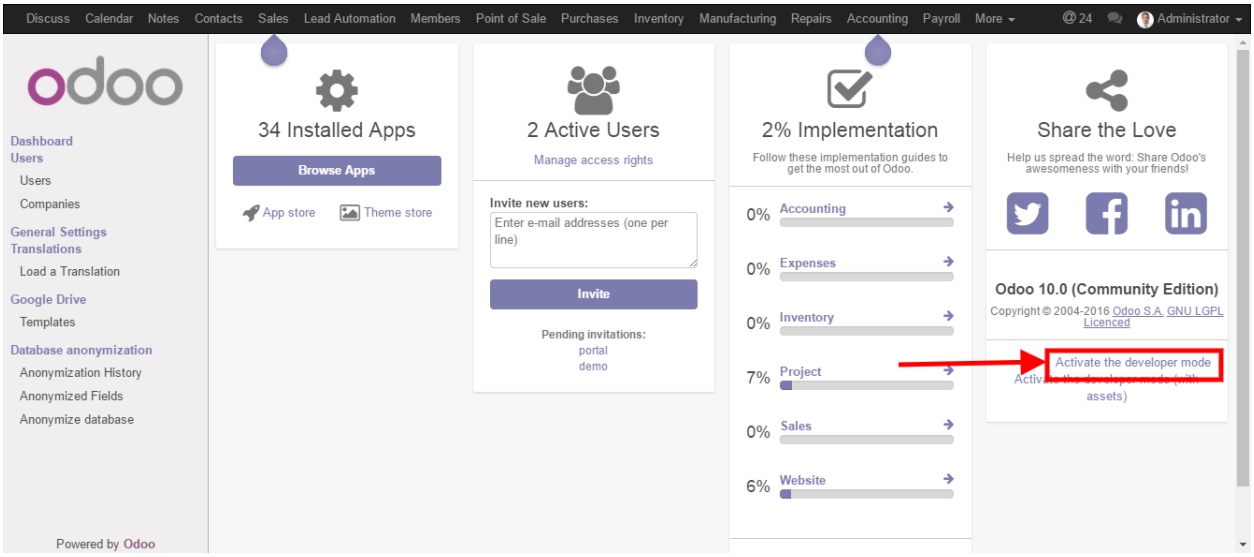
13.0+

- перейти к “ Настройки“
- go to `General Settings`
- scroll to **Developer Tools** section



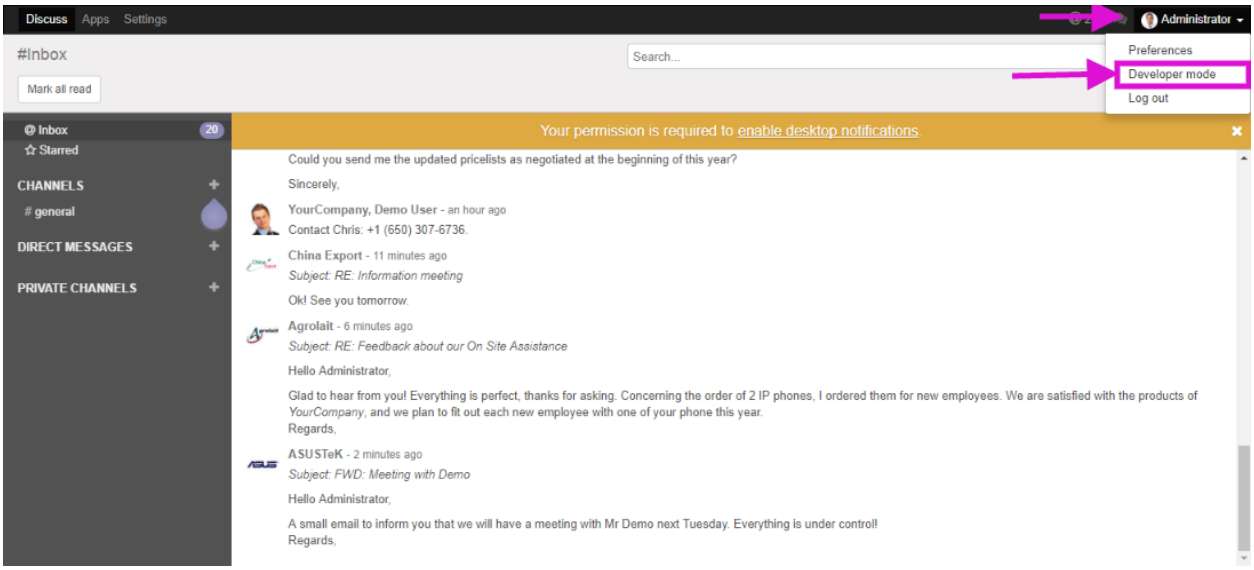
10.0, 11.0, 12.0

- перейти к “ Настройки“
- нажмите “ Активировать режим разработчика“



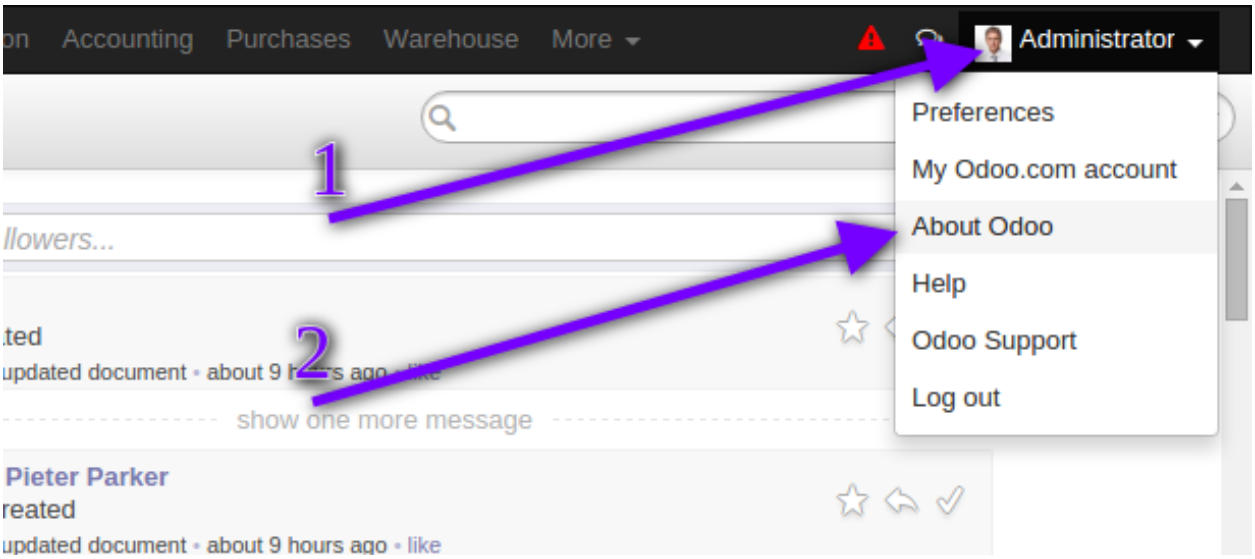
10.0, 11.0, 12.0 with the web_debranding

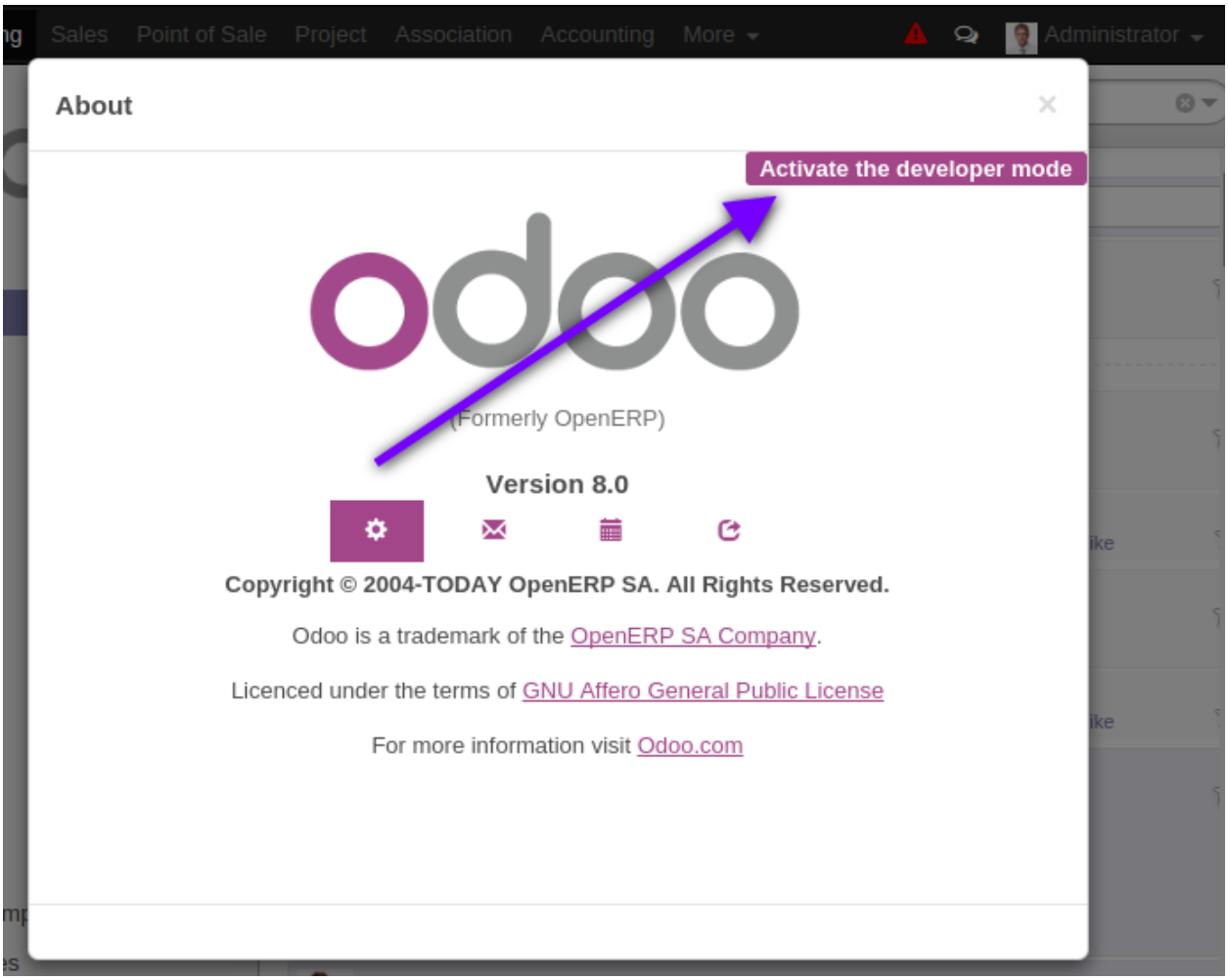
- перейдите в меню пользователя в правом верхнем углу
- нажмите “ Режим разработчика“



9,0, 8,0

- нажмите кнопку в правом верхнем углу “ <User Name> -> О Одо“
- нажмите “ Активировать режим разработчика“





- В odoo 8.0 вам может понадобиться :doc:‘Enable technical features<technical-features>‘тоже

8.2.5 Как активировать режим отладочных ресурсов

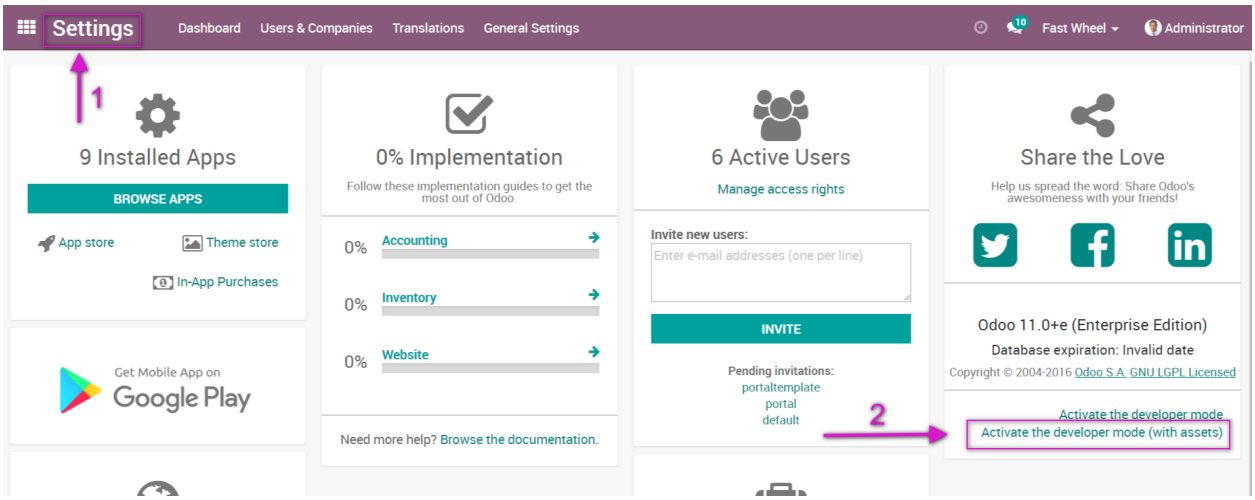
Добавьте параметр “ debug = assets“ в ваш URL, например:

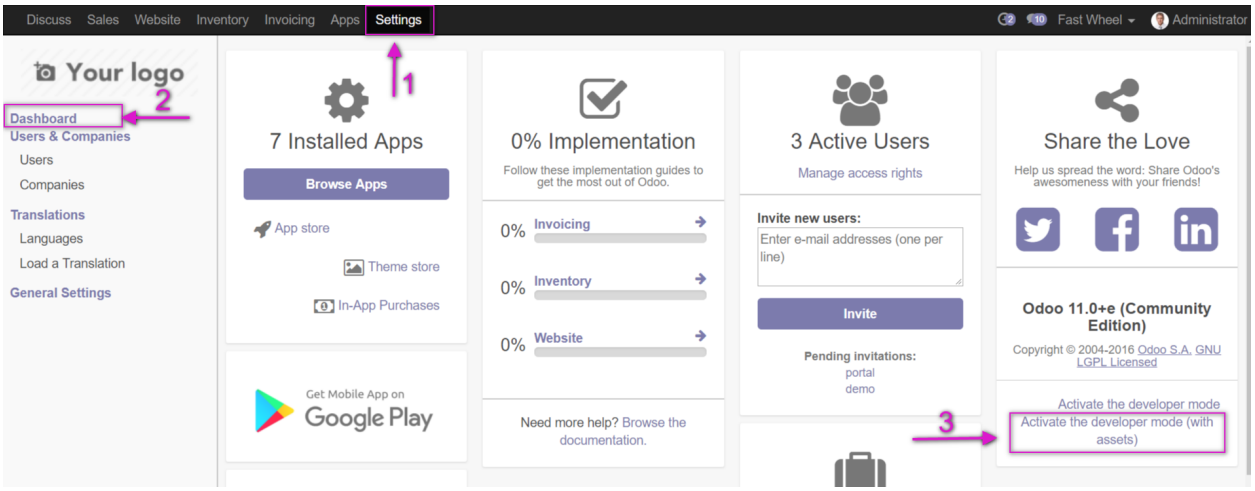
```
localhost:8069/web?debug=assets#
```

или используйте пользовательский интерфейс, как описано ниже

10.0+

- перейти к “ Настройки“
- нажмите “ Активировать режим разработчика (с активами) “





8.2.6 Как войти в систему как суперпользователь

12.0+

- открыть страницу с формой входа, например:

```
localhost:8069/web/login
```

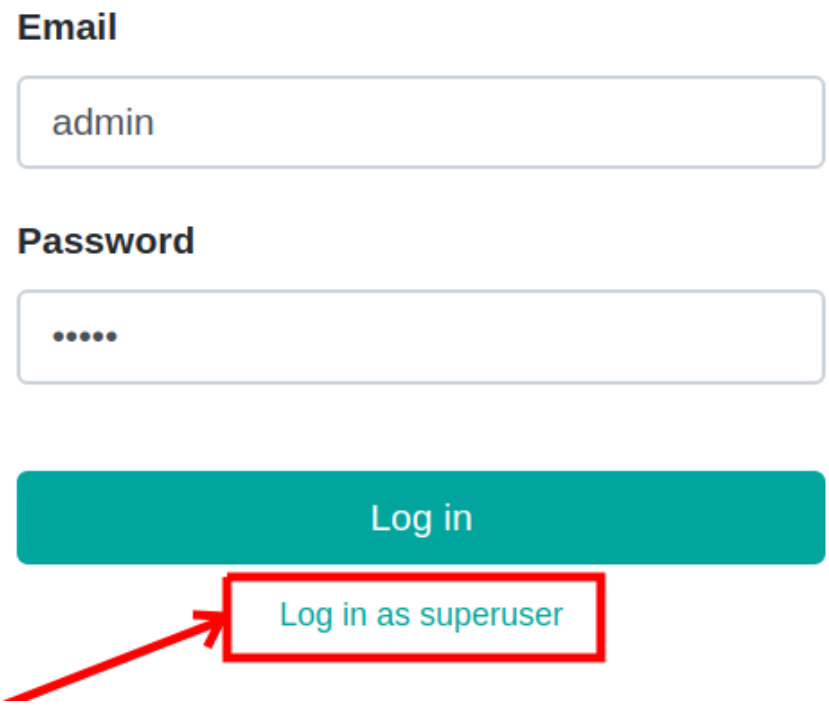
- добавьте параметр “ debug“ в ваш URL, например:

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

- введите имя пользователя и пароль пользователя, который входит в группу безопасности “ Administration: Settings“ (“ base.group_system“), например:

```
Username: admin
Password: admin
```

- нажмите на ссылку “ Войти как суперпользователь“ под кнопкой “ Войти в систему“



Глава 9

Администрация Odoo

Официальные документы:

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

9.1 Как включить Longpolling в odoo

Longpolling - это способ доставки мгновенного уведомления веб-клиенту (например, в чатах).

Чтобы активировать лонгполлинг:

- установить зависимости
  - odoo 11.0

```
python -c "import gevent" || sudo pip3 install gevent
```

  - odoo 10.0

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

- установить ненулевое значение для :doc:‘workers <workers>‘параметр
- настроить nginx

```
location /longpolling {
    proxy_pass http://127.0.0.1:8072;
}
location / {
    proxy_pass http://127.0.0.1:8069;
}
```

- если вы устанавливаете odoo 9.0 через пакет deb, вам нужно восстановить файл openerp-gevent (см. # 10207):

```
cd /usr/bin/
wget https://raw.githubusercontent.com/odoo/odoo/9.0/openerp-gevent
chmod +x openerp-gevent
```

Узнайте больше о longpolling

9.2 Про лонгполлинг

** Что такое длинный опрос HTTP? **

Веб-приложения изначально разрабатывались на основе модели клиент / сервер, где клиент всегда является инициатором транзакций, запрашивающих данные с сервера. Таким образом, у сервера не было механизма для самостоятельной отправки или передачи данных клиенту без предварительного запроса клиента.

** В двух словах: HTTP длинный опрос **

Чтобы преодолеть этот недостаток, разработчики веб-приложений могут реализовать метод, называемый длинным опросом HTTP, при котором клиент опрашивает сервер, запрашивая новую информацию. Сервер удерживает запрос открытым до тех пор, пока не будут доступны новые данные. Когда сервер станет доступен, он ответит и отправит новую информацию. Когда клиент получает новую информацию, он немедленно отправляет другой запрос, и операция повторяется. Это эффективно имитирует функцию проталкивания сервера.

Таким образом, каждый пакет данных означает новое соединение, которое будет оставаться открытым, пока сервер не отправит информацию.

На практике соединение обычно переустанавливается один раз в 20-30 секунд, чтобы избавиться от возможных проблем (ошибок), например, проблем, связанных с HTTP-прокси.

В отличие от обычного опроса, такое уведомление появляется быстрее.

‘‘ Задержка = установка соединения + передача данных‘‘

** Преимущества лонгполлинга **

- Загрузка на сервер снижена в отличие от обычного опроса
- Снижение трафика
- Поддержка во всех современных браузерах

Таким образом, длинный опрос помогает клиенту получать данные, как только они появляются на сервере, в отличие от периодических, которые отправляют запросы в соответствии с указанным интервалом.

9.3 ‘‘ –Workers‘‘

Based on this comment from Odony: https://github.com/odoo/odoo/issues/39825#issuecomment-555256475

Таким образом, в документации говорится, что один работник может обслуживать 6 пользователей. Это означает, что работник может обрабатывать в среднем ~ 6 тяжелых операций чтения в секунду (150 мс каждая) = 6 веб-запросов / с. Если пользователь активирует около 60 тяжелых запросов в

минуту во время активного использования, это в среднем 1 требование / с, поэтому 6 пользователей могут максимально использовать работника во время пиков, когда все они активны. Но на самом деле люди не создают устойчивую нагрузку, и реальное использование со временем будет в среднем намного меньше, может быть, 20% от этого, так что один работник может справиться с десятками обычных пользователей. Если вы не сталкиваетесь с патологическими случаями, например, с классом, в котором все ученики нажимают одновременно, или с тяжелыми автоматизированными сценариями RPC (не-тяжелые пользователи), вы можете начать с 1 работника на 30 пользователей, может даже 40 на нескольких случай арендатора, когда пользователи распределены по разным часовым поясам, и не все базы данных активны одновременно.

Если вы не знаете, сколько рабочих вам потребуется, начните с 10, но постарайтесь иметь гибкость (в ОЗУ и ЦП) для более удобного развертывания по мере необходимости. Мониторинг вашей системы, чтобы увидеть, как вы делаете с точки зрения ресурсов и скорости транзакций.

Другие вещи для рассмотрения:

- Всегда настраивайте более 6 рабочих, так как браузеры должны будут открывать много параллельных соединений, и вы не хотите, чтобы они ставились в очередь, так как пользователи будут чувствовать задержки. 6 или 8 - минимум, даже если у вас недостаточно процессоров.
- Реальное ограничение количества рабочих - это оперативная память, а не процессоры. Если рабочие x limit_memory_hard намного больше доступной оперативной памяти, вы можете вызвать обмен или сбой. В наши дни, по крайней мере, 32 ГБ или 64 ГБ ОЗУ, это немного, и если вы не выделите все для Odoo, остальное будет полезно для кэша ОС и буферов.
- Вы можете использовать 2 x num_cpus + 1 работника, чтобы убедиться, что вы будете использовать все доступные ядра. Имея меньше работников, чем это трата ресурсов. Но вы можете иметь больше рабочих, если хотите, если у вас достаточно оперативной памяти.
- Скорость процессора имеет значение, поэтому постарайтесь получить максимальную тактовую частоту процессора, какую только сможете. Лучше разделить рабочих на несколько серверов с меньшим количеством процессорных ядер, но с более высокой тактовой частотой.

9.3.1 wkhtmltopdf

Workers value must be at least 2 to make wkhtmltopdf work.

9.3.2 Longpolling

Скрытой особенностью Multiprocessing является автоматический запуск процесса gevent для поддержки longpolling.

Longpolling - это дополнительный процесс, т.е. если у вас “ –workers = 2“, вы получите 2 рабочих процесса и 1 процесс gevent

9.4 “ –Db_maxconn“

Вот определение из “ 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")
```

Более точное объяснение этой опции следующее:

“ db_maxconn“ - указать максимальное количество физических подключений к posgresql ** на процесс odoo, но для всех баз данных **

** Сколько процесс ODOO работает? **

- : Документ: ‘longpolling <longpolling> ‘- не более 1 процесса
- : Doc: ‘рабочие <workers> ‘
- : Doc: ‘max_cron_threads <max_cron_threads> ‘

** Что это означает практически? **

Если у вас развертывание с большим количеством баз данных или одновременных пользователей, вы можете столкнуться со следующей ошибкой:

```
File "/opt/odoo/vendor/odoo/cc/odoo/service/wsgi_server.py", line 128, in application
    return application_unproxied(environ, start_response)
File "/opt/odoo/vendor/odoo/cc/odoo/service/wsgi_server.py", line 117, in application_unproxied
    result = odoo.http.root(environ, start_response)
File "/opt/odoo/vendor/odoo/cc/odoo/http.py", line 1331, in __call__
    return self.dispatch(environ, start_response)
File "/opt/odoo/vendor/odoo/cc/odoo/http.py", line 1300, in __call__
    return self.app(environ, start_wrapped)
File "/opt/odoo/.local/lib/python3.7/site-packages/werkzeug/wsgi.py", line 766, in __call__
    return self.app(environ, start_response)
File "/opt/odoo/vendor/odoo/cc/odoo/http.py", line 1501, in dispatch
    result = ir_http._dispatch()
File "/opt/odoo/vendor/odoo/cc/addons/auth_signup/models/ir_http.py", line 19, in _dispatch
    return super(Http, cls)._dispatch()
File "/opt/odoo/vendor/odoo/cc/addons/web_editor/models/ir_http.py", line 22, in _dispatch
    return super(IrHttp, cls)._dispatch()
File "/opt/odoo/vendor/odoo/cc/odoo/addons/base/models/ir_http.py", line 207, in _dispatch
    return cls._handle_exception(e)
File "/opt/odoo/vendor/odoo/cc/odoo/addons/base/models/ir_http.py", line 174, in _handle_exception
    raise exception
File "/opt/odoo/vendor/odoo/cc/odoo/addons/base/models/ir_http.py", line 203, in _dispatch
    result = request.dispatch()
File "/opt/odoo/vendor/odoo/cc/odoo/http.py", line 840, in dispatch
    r = self._call_function(**self.params)
File "/opt/odoo/vendor/odoo/cc/odoo/http.py", line 351, in _call_function
    return checked_call(self.db, *args, **kwargs)
File "/opt/odoo/vendor/odoo/cc/odoo/service/model.py", line 97, in wrapper
    return f(dbname, *args, **kwargs)
File "/opt/odoo/vendor/odoo/cc/odoo/http.py", line 344, in checked_call
    result = self.endpoint(*a, **kw)
File "/opt/odoo/vendor/odoo/cc/odoo/http.py", line 946, in __call__
    return self.method(*args, **kw)
File "/opt/odoo/vendor/odoo/cc/odoo/http.py", line 524, in response_wrap
    response = f(*args, **kw)
File "/opt/odoo/vendor/odoo/cc/addons/auth_signup/controllers/main.py", line 21, in web_login
    response = super(AuthSignupHome, self).web_login(*args, **kw)
File "/opt/odoo/vendor/odoo/cc/odoo/http.py", line 524, in response_wrap
    response = f(*args, **kw)
File "/opt/odoo/vendor/odoo/cc/addons/web/controllers/main.py", line 484, in web_login
    values['databases'] = http.db_list()
File "/opt/odoo/vendor/odoo/cc/odoo/http.py", line 1517, in db_list
    dbs = odoo.service.db.list_dbs(force)
File "/opt/odoo/vendor/odoo/cc/odoo/service/db.py", line 379, in list_dbs
    with closing(db.cursor()) as cr:
```

(continues on next page)

(продолжение с предыдущей страницы)

```
File "/opt/odoo/vendor/odoo/cc/odoo/sql_db.py", line 657, in cursor
    return Cursor(self.__pool, self.dbname, self.dsn, serialized=serialized)
File "/opt/odoo/vendor/odoo/cc/odoo/sql_db.py", line 171, in __init__
    self._cnx = pool.borrow(dsn)
File "/opt/odoo/vendor/odoo/cc/odoo/sql_db.py", line 540, in _locked
    return fun(self, *args, **kwargs)
File "/opt/odoo/vendor/odoo/cc/odoo/sql_db.py", line 608, in borrow
    **connection_info)
File "/usr/local/lib/python3.7/site-packages/psycopg2/__init__.py", line 130, in connect
    conn = _connect(dsn, connection_factory=connection_factory, **kwasync)
psycopg2.OperationalError: FATAL: sorry, too many clients already
```

Для ее решения необходимо настроить следующие параметры:

- В оду
  - “ db_maxconn“
  - : Doc: ‘рабочие <workers> ‘
  - : Doc: ‘max_cron_threads <max_cron_threads> ‘
- В posgresql
  - max_connections

Эти параметры должны удовлетворять следующему условию:

```
(1 + workers + max_cron_threads) * db_maxconn < max_connections
```

Например, если у вас есть следующие значения:

- работники = 1 (минимальное значение, чтобы заставить работать долго)
- max_cron_threads = 2 (по умолчанию)
- db_maxconn = 64 (по умолчанию)
- max_connections = 100 (по умолчанию)

затем “ (1 + 1 + 2) * 64 = 256> 100“, т.е. условие не выполняется, и такое развертывание может столкнуться с ошибкой, описанной выше.

** Хорошо, но какие значения хороши для конкретного сервера и условий нагрузки? **

> Оформить заказ <https://github.com/odoo/odoo/issues/39825#issuecomment-555175814>‘__ из одони. В частности, для параметра “ db_maxconn“ цитата приведена ниже.

PostgreSQL's `max_connections` should be set higher than `db_maxconn * number_of_processes`. You may need to tweak the kernel sysctl if you need `max_connections` higher than 1-2k.

В режиме многопроцессорной обработки каждый рабочий HTTP обрабатывает один запрос за раз, поэтому теоретически может работать “ db_maxconn = 2“ (для некоторых запросов требуется 2 курсора, следовательно, 2 дБ соединения). Однако для мультитенанта это не оптимально, потому что каждый запрос должен будет заново открывать новое соединение с другой базой данных - лучше установить его немного выше. С большим количеством работников 32 - это хороший компромисс, так как 64 может заставить вас достичь ограничений ядра. Также имейте в виду, что ограничение применяется и к работнику с длинным опросом, и вы не хотите слишком долго откладывать сообщения чата из-за полного пула соединений, поэтому не устанавливайте его слишком низким, несмотря ни на что. Сохранение значения в диапазоне 32-64 обычно кажется хорошим выбором.

Для многопоточного режима, поскольку существует только 1 процесс, это размер глобального пула соединений. Во избежание ошибок его следует устанавливать между 1x и 2x ожидаемым количеством одновременных запросов одновременно. Можно оценить на основе количества баз данных и ожидаемой активности. Если один процесс обрабатывает более 20 запросов одновременно на одном ядре (помните, что многопотоковость зависит от GIL) вряд ли даст хорошую производительность, поэтому, опять же, настройка в диапазоне 32-64, скорее всего, будет работать для нормальная нагрузка.

9.5 “ –Max-хрон-threads“

Вот определение из “ 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")
```

9.6 --addons-path

9.6.1 Дублирующиеся аддоны

Если у вас есть две папки с одним и тем же модулем, и у вас есть причина добавить обе папки в “ addons_path“, то будет использоваться первая найденная версия модуля. То есть папка в начале списка “ addons_path“ имеет больший приоритет.

9.7 “ –Log-handler“

```
--log-handler=PREFIX:LEVEL
```

Устанавливает обработчик на УРОВЕНЬ для данного ПРЕФИКСА. Эта опция может быть повторена.

For example, if you want to have DEBUG level for module sync only, you can run it with parameter:

```
--log-handler=odoo.addons.sync:DEBUG
```

Чтобы отключить логи werkzeug, добавьте следующий параметр:

```
--log-handler=werkzeug:CRITICAL
```

Чтобы увидеть все сообщения журнала odoo:

```
--log-handler=odoo:DEBUG
```

Чтобы увидеть все сообщения журнала (в том числе от libs):

```
--log-handler=:DEBUG
```

9.7.1 Уровни журнала

- КРИТИЧЕСКИЕ

- ОШИБКА
- ПРЕДУПРЕЖДЕНИЕ
- ИНФОРМАЦИЯ
- DEBUG
- НЕ УСТАНОВЛЕНО

9.7.2 Полезные журналы

Показать запросы API:

```
--log-handler=odoo.api:DEBUG
```

9.7.3 Использование в конфигурационном файле

Чтобы выполнить настройки через конфигурационный файл, используйте ключевое слово ‘‘ log_handler‘‘ и задайте значения в виде списка, разделенного запятыми, например:

```
log_handler=werkzeug:CRITICAL,odoo.api:DEBUG
```

9.8 ‘‘ –Db-filter‘‘

Основная цель *–db-filter‘* - не спрашивать пользователя, какую базу данных ему нужно использовать (он может этого не знать). Это реализуется путем проверки адреса HOST, который был использован.

Например, у вас есть два независимых веб-сайта, например, ‘‘ shop1.example.com‘‘ и ‘‘ shop2.example.com‘‘, которые указывают на один и тот же сервер odoo с двумя базами данных. Используя ‘‘ –db-filter‘‘, вы можете настроить odoo на использование соответствующей базы данных в зависимости от используемого адреса хоста. Проверьте ссылки на документацию ниже или перейдите к примерам, чтобы узнать, как это сделать.

9.8.1 Документы

Официальная документация: https://www.odoo.com/documentation/master/setup/deploy.html#dbfilter

Основной код: https://github.com/odoo/odoo/search?l=Python&q=%22def+db_monodb%22

Дополнительная опция: https://github.com/OCA/server-tools/tree/11.0/dbfilter_from_header

9.8.2 Примеры

Единая база данных

Если у вас есть одна база данных, вы можете установить фильтр по умолчанию:

```
--db-filter=.*
```

Игнорирование других баз данных

Чтобы заставить odoo всегда использовать только одну базу данных, скажем “ mydb“, используйте следующий фильтр

```
--db-filter=^mydb$
```

Имена баз данных равны имени хоста

```
--db-filter=^%h$
```

Чтобы использовать фильтр выше, вы должны назвать базы данных равными адресу хоста, например:

- “ shop1.example.com“ - имя первой базы данных
- “ shop2.example.com“ - имя второй базы данных
- “ www.super-shop.example.com“ - название третьей базы данных
- “ it-projects.info“ - название четвертой базы данных

Предупреждение: этот фильтр не может работать с префиксом “ www“ и без него одновременно

Имена баз данных, равные поддомену

```
--db-filter=^%d$
```

Чтобы использовать фильтр выше, вы должны назвать базы данных равными поддомену, например, если имя базы данных равно “ shop“, тогда фильтр будет использовать его для любого из следующих запросов:

- “ shop.example.com“
- “ Www.shop.example.com“
- “ shop.yourbrand.example“
- “ Www.shop.yourbrand.example“

9.9 “ –Load“

Параметр “ –load“ (также известный как “ server_wide_modules“) используется для определения списка модулей, которые загружаются при запуске odoo. Такие модули загружаются, даже если в odoo нет баз данных. Odoo по умолчанию загружает модуль “ web“, но точный список может отличаться для разных версий odoo

9.9.1 Значение по умолчанию

- Odoo 15: `base,web`
- Odoo 14: `base,web`
- Odoo 13: `base,web`

- Odoo 12:‘ *base, web*‘
- Odoo 11:‘ *web*‘
- Odoo 10:‘ *web, web_kanban*‘
- Odoo 9:‘ *web, web_kanban*‘
- Odoo 8:‘ *web, web_kanban*‘
- Odoo 7:‘ *web, web_kanban*‘

9.9.2 Добавление нового модуля в список

В общем, вам нужно взять значение по умолчанию и добавить туда новый модуль.

Через CLI

```
# Example for Odoo 12.0:
./odoo-bin --workers=2 --load base,web,NEWMODULE --config=/path/to/odoo.conf
```

Через конфигурационный файл

Имя параметра в файле конфигурации: “ server_wide_modules“. Добавьте этот параметр, если он еще не представлен, или измените существующее значение, добавив новый модуль

```
[options]
# example for Odoo 12.0:
server_wide_modules=base,web,NEWMODULE
# ...
```

Odoo.sh

- перейдите на вкладку Shell в odoo.sh
- выполнить “ nano .config / odoo / odoo.conf“
- добавить параметр “ server_wide_modules“ с добавлением NEWMODULE (см. выше)
- перезапустите сервер, выполнив следующую команду: “ odoosh-restart“

9.10 PosBox

Официальные документы:

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

9.10.1 Запуск PosBox на вашем компьютере в целях разработки

Запуск PosBox на вашем компьютере означает запуск второго сервера odoo вместо PosBox.

Для запуска второго сервера odoo необходимо изменить параметры конфигурации, которые отличаются от текущих настроек первого сервера odoo.

Для этого просто измените значение порта “ xmlrpc“ и “ longpolling“.

Например, если настройки запуска для первого odoo-сервера “ / path / to / openerp-server1.conf“

```
xmlrpc_port = 8069
longpolling_port = 8072
```

тогда настройки для второго odoo-сервера “ / path / to / openerp-server2.conf“ могут быть следующими:

```
xmlrpc_port = 9069
longpolling_port = 9072
```

Пример запуска ** PosBox ** на вашем компьютере с использованием “ Network Printer“:

- Запустите первый сервер Odoo, например:

  ```
  ./openerp-server --config=/path/to/openerp-server1.conf
  ```

- Установите Pos Printer Network модуль на Odoo в‘ обычном <http://odoo-development.readthedocs.io/en/latest/odoo/usage/install-module.html?highlight=install#from-app-store-install>‘_ способ.
- Сконфигурируйте PosBox, используя инструкции по установке.
- Запустите второй Odoo Server, используя новые настройки, и добавьте в “ –load parameters“, например:

  ```
  ./openerp-server --load=web,hw_proxy,hw_posbox_homepage,hw_scale,hw_scanner,hw_escpos,hw_
  ↪printer_network --config=/path/to/openerp-server2.conf
  ```

- Печать в сетевом принтере.

Запустите PosBox через докер

Пример с hw_printer_network и‘ PosBox 8.0 <https://github.com/odoo/odoo/tree/8.0/addons/point_of_sale/tools/posbox>‘_:

```
docker run -d -p 1984:1984 --name wdb kozea/wdb
docker run -d -e POSTGRES_USER=odoo -e POSTGRES_PASSWORD=odoo --name db-posbox-8.0 postgres:9.5

docker run \
-p 9069:8069 \
-p 9072:8072 \
--link wdb:wdb -e WDB_SOCKET_SERVER=wdb -e WDB_NO_BROWSER_AUTO_OPEN=True \
-e 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
```

Чтобы использовать вашу версию встроенных модулей odoo, используйте следующий “ -v путь / к / odoo: / mnt / odoo-source“.

Источник этого докера можно найти здесь: https://github.com/it-projects-llc/install-odoo/tree/8.0/dockers/posbox

Предупреждение: Это на самом деле не работает и выдает ошибку «Нет доступного бэкэнда». Вероятно, вместо - *–privileged -v / dev / bus / usb: / dev / bus / usb* ‘следует использовать –device = / dev / SOMETHING

9.10.2 Установка PosBox

Скачать последнюю версию “ posbox_image“:

- https://nightly.odoo.com/master/posbox/

Примечание: Используйте другой компьютер с устройством для чтения карт SD, чтобы установить образ.

Вам нужно будет использовать инструмент для записи изображений, чтобы установить изображение, которое вы загрузили на свою SD-карту.

** Etcher ** - это графический инструмент для записи на SD-карту, который работает в Mac OS, Linux и Windows и является самым простым вариантом для большинства пользователей. Etcher также поддерживает запись изображений непосредственно из zip-файла без необходимости распаковки. Чтобы написать свое изображение с Etcher:

- Скачать Etcher и установите его.
- Подключите устройство для чтения SD-карт к SD-карте внутри.
- Откройте Etcher и выберите на своем жестком диске файл Raspberry Pi “ .img“ или “ .zip“, который вы хотите записать на SD-карту.
- Выберите SD-карту, на которую вы хотите записать свое изображение.
- Проверьте свой выбор и нажмите «Flash!» начать запись данных на SD-карту.

** Подключение периферийных устройств **

Официально поддерживаемое оборудование указано на странице «POS Hardware». <https://www.odoo.com/page/point-of-sale-hardware>‘_, но другое оборудование также может работать.

- “ Принтер: “ Подключите принтер ESC / POS к порту USB и включите его.
- “ Денежный ящик: “ Денежный ящик должен быть подключен к принтеру с помощью кабеля RJ25.
- “ Сканер штрих-кода: “ Подключите ваш сканер штрих-кода. Для того, чтобы ваш сканер штрих-кода был совместимым, он должен вести себя как клавиатура и должен быть настроен в США. QWERTY. Он также должен заканчиваться штрих-кодами с символа Enter (код клавиши 28). Скорее всего, это стандартная конфигурация вашего сканера штрих-кода.
- “ Весы: “ Подключите весы и включите их.
- “ Ethernet: “ Если вы не хотите использовать Wi-Fi, подключите кабель Ethernet. Убедитесь, что это подключит POSBox к той же сети, что и ваше POS-устройство.

- “ Wi-Fi: “ Если вы не хотите использовать Ethernet, подключите совместимый с Linux USB-адаптер Wi-Fi. Большинство имеющихся в продаже адаптеров Wi-Fi совместимы с Linux. Официально поддерживаются адаптеры Wi-Fi с чипсетом Ralink 5370. Не подключайте кабель Ethernet, потому что все функции Wi-Fi будут отключены, если доступно проводное сетевое соединение.
- “ Сетевой принтер: “ Подключить сетевой принтер.

** Питание POSBox **

Подключите адаптер питания к POSBox, должен загореться ярко-красный индикатор состояния.

** Убедитесь, что POSBox готов **

После включения POSBox требуется некоторое время для загрузки. Когда POSBox готов, он должен распечатать квитанцию о состоянии со своим IP-адресом. Также индикатор состояния, расположенный рядом с красным индикатором питания, должен постоянно гореть зеленым цветом.

Больше информации читайте в:

- https://www.raspberrypi.org/documentation/installation/installing-images/
- https://www.odoo.com/documentation/user/9.0/point_of_sale/overview/setup.html

9.10.3 Вступление

** POSBox ** выполняет сильно модифицированную установку ** Raspbian Linux **, производную Debian для ** Raspberry Pi **. Он также запускает установку Odoo, которая предоставляет веб-сервер и драйверы. Аппаратные драйверы реализованы в виде модулей Odoo. Все эти модули имеют префикс “ **hw_** * “ и являются единственными модулями, работающими на POSBox. Odoo используется только для платформы, которую он предоставляет. Бизнес-данные не обрабатываются и не хранятся в POSBox. Экземпляр Odoo - это мелкий клон git ветки “ 8.0“.

Корневой раздел на POSBox монтируется «только для чтения», поэтому мы не изнашиваем SD-карту, слишком много записывая на нее. Это также гарантирует, что файловая система не может быть повреждена путем отключения питания POSBox. Приложения Linux ожидают, что смогут писать в определенные каталоги. Поэтому мы предоставляем виртуальный диск для “ / etc“ и “ / var“ (Raspbian автоматически предоставляет диск для “ / tmp“). Эти виртуальные диски настраиваются с помощью “ setup_ramdisks.sh“, который мы запускаем перед всеми другими сценариями инициализации, запустив его в “ / etc / init.d / rcS“. RAM-диски называются “ / etc_ram“ и “ / var_ram“ соответственно. Большая часть данных из “ / etc“ и “ / var“ копируется в эти виртуальные диски tmpfs. Чтобы ограничить размер виртуальных дисков, мы не копируем для них определенные вещи (например, соответствующие данные). Затем мы связываем их поверх оригинальных каталогов. Поэтому, когда приложение пишет в “ / etc / foo / bar“, оно фактически записывает в “ / etc_ram / foo / bar“. Мы также привязываем “ / “ к “ / root_bypass_ramdisks“, чтобы иметь возможность добраться до реальных “ / etc“ и “ / var“ во время разработки.

9.10.4 Как редактировать конфиг

Если у вас есть IP-адрес POSBox и SSH-клиент, вы можете получить удаленный доступ к системе POSBox.

** Логин: ** “ pi“ ** Пароль: ** “ Малина“

Помните, что root (/) монтируется только для чтения, поэтому вы не можете использовать запись.

Если вы хотите использовать его, вам нужно перезагрузиться в обычном режиме.

```
sudo su
mount -o rw,remount /
mount -o rw,remount /root_bypass_ramdisks
```

синхронизировать и перезагрузить posbox

```
sync
reboot
```

9.10.5 Как обновить параметры командной строки odoo

edit /root_bypass_ramdisks/etc/init.d/odoo

```
nano /root_bypass_ramdisks/etc/init.d/odoo
```

добавьте “ hw_printer_network“ в “ –load параметр“

```
$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
```

9.10.6 Как редактировать исходники odoo

Закомментируйте строку 354 в “ hw_escpos / controllers / main.py“

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

т.е. заменить “ driver.push_task (’printstatus’) “ на

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

синхронизировать и перезагрузить posbox

```
sync
reboot
```

9.10.7 Чтение логов на posbox

Чтение журналов

```
tail -f /var/log/odoo/odoo-server.log
```

Редактировать уровень журнала:

```
nano /home/pi/odoo/addons/point_of_sale/tools/posbox/configuration/odoo.conf
```

заменить на

```
log_level = info
```

Глава 10

Непрерывная доставка

СДЕЛАТЬ

Глава 11

техническое обслуживание

11.1 Перенос данных

Перенос данных - это процесс сохранения правильных данных в базе данных после обновления до новой версии модуля. Например, простое переименование полей приводит к потере данных, если у вас нет соответствующих сценариев миграции данных.

Для * Миграции модулей * см . *Porting Modules*

Примечание: Module porting and data migration combined is usually called *Database Upgrading* in Odoo world

11.1.1 Подготовка

Эти миграции между версиями модуля.

От Odoo https://github.com/odoo/odoo/blob/11.0/odoo/modules/migration.py#L53:

Этот класс управляет миграцией модулей. Файлы миграции должны быть python-файлами, содержащими функцию ‘‘ migrate (cr, instal_version) ‘‘. Эти файлы должны соответствовать структуре дерева каталогов: папка ’migrations’, которая содержит папку по версии. Версия может быть версией ’module’ или версией ’server.module’ (в этом случае файлы будут обрабатываться только этой версией сервера). Имена файлов Python должны начинаться с * pre * или * post * и будут выполняться соответственно до и после инициализации модуля. Сценарии * end * запускаются после обновления всех модулей.

Пример:

```
<moduledir>
`-- migrations
    |-- 1.0
    |   |-- pre-update_table_x.py
    |   |-- pre-update_table_y.py
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    |   |-- post-create_plop_records.py
    |   |-- end-cleanup.py
    |   `-- README.txt                      # not processed
    |-- 9.0.1.1                             # processed only on a 9.0 server
    |   |-- pre-delete_table_z.py
    |   `-- post-clean-data.py
    `-- foo.py                              # not processed
```

11.1.2 выполнение

Миграционные файлы - это просто файлы кода, которые не нужно нигде регистрировать. При обновлении аддона Odoo выполняет поиск в * миграциях * папок с версией между ними, включая версию, для которой выполняется обновление. Это происходит до того, как все остальные файлы были обнаружены, поэтому в данный момент ничего не меняется в вашей структуре базы данных. Затем, если папки найдены, Odoo запускает файлы python с префиксом * pre- *. Они должны содержать определенную функцию с именем migrate. Эта функция имеет два аргумента: курсор базы данных и текущую установленную версию.

После того, как все функции перед миграцией были успешно выполнены, Odoo обновляет модуль. Теперь база данных может отличаться от предыдущей версии. Например, если в новой версии мы изменили тип поля модели, в базе данных этот столбец будет изменен без сохранения данных. Или, если поле было переименовано, в новой версии будет создан только новый столбец.

Затем, после обновления модуля, Odoo ищет файлы после переноса по тому же алгоритму и выполняет их.

Сценарии * end * запускаются после обновления всех модулей.

Предупреждение: Обновления миграции не отменяются, если некоторые ошибки произошли позже во время обновления модулей. Таким образом, вы всегда должны сначала пытаться обновить модуль с помощью скриптов миграции на копии.

11.1.3 пример

POS Debt & Credit notebook. We need to preserve credit_product field data in the `product.template` model after updating to a newer version. In previous version it was boolean field, now it is a many2one field with the relation to `account.journal` model. Here, we, using a temporary column, calculate transfer data from boolean to many2one credit_product field.

pre-migrate.py:

```
def migrate(cr, version):
    # Add temporary credit product column
    cr.execute('ALTER TABLE product_template ADD temporary_credit_product int')
    # Select debt journals
    cr.execute('SELECT id FROM account_journal WHERE account_journal.debt is true')
    # Take the first journal
    journal_id = cr.fetchone()
    if journal_id:
        # set token one to all credit products
        cr.execute('UPDATE product_template SET temporary_credit_product=%s WHERE credit_product␣
↪is true', journal_id)
```

post-migrate.py:

```
def migrate(cr, version):
    # update new credit_product column with the tempory one
    cr.execute('UPDATE product_template SET credit_product=temporary_credit_product')
    # Drop temporary column
    cr.execute('ALTER TABLE product_template DROP COLUMN temporary_credit_product')
```

Глава 12

IDE

12.1 Emacs

12.1.1 Emacs

Установите emacs 24.4+ http://askubuntu.com/questions/437255/how-to-install-emacs-24-4-on-ubuntu

- Открытые Emacs
- Нажмите Alt-x пакет-список-пакетов
- установить пакеты: нажмите i, а затем x
- некоторые пакеты требуют зависимостей, которые должны быть установлены через терминал * flymake * loccur * flymake-css * flymake-jslint * flymake-python-pyflakes

```
sudo pip install flake8
```

  - Magit
  - СП3-режим

12.1.2 Spacemacs

Требования

- Emacs версия 24 или новее.

Установка

Установите spacemacs из github https://github.com/syl20bnr/spacemacs

Документация

http://spacemacs.org/doc/DOCUMENTATION.html

Слои для разработки Odoo

Используйте следующие слои:

- автозавершение
- лучше по умолчанию
- Emacs-шепелявость
- мерзавец
- Синтаксис проверка
- контроль версий
- pyhton
- eyebrowse
- SQL
- питон
- семантический

Проверка синтаксиса в python использует пакет pylint (http://liuluheng.github.io/wiki/public_html/Python/flycheck-pylint-emacs-with-python.html). Установите его

```
sudo pip install pylint
```

12.1.3 Заменить текст в рекурсивно найденных файлах

1. ‘‘ Mx find-name-dired‘‘: вам будет предложено указать корневой каталог и шаблон имени файла.
2. Нажмите ‘‘ t‘‘, чтобы «пометить метку» для всех найденных файлов.
3. Нажмите ‘‘ Q‘‘ для «Query-Replace in Files . . . »: вам будет предложено указать регулярные выражения запроса / замены.
4. Действуйте как с ‘‘ query-replace-regexp‘‘: ‘‘ SPACE‘‘ для замены и перехода к следующему совпадению, ‘‘ n‘‘ для пропуска совпадения и т. Д.

Основано на: http://stackoverflow.com/questions/270930/using-emacs-to-recursively-find-and-replace-in-text-files-not-alrea

12.1.4 Pylint

Pylint - это инструмент, который проверяет ошибки в коде Python, пытается внедрить стандарт кодирования и ищет запахи кода. Он также может искать определенные ошибки типа, он может порекомендовать предложения о том, как конкретные блоки могут быть реорганизованы, и может предложить вам подробности о сложности кода. https://pylint.readthedocs.io/en/latest/

Установить пилинт.

```
sudo pip install pylint
```

С расширением Flycheck emacs вывод pylint будет показан прямо внутри ваших буферов emacs. Spacemacs имеет flycheck в своем слое “ синтаксическая проверка“.

```
M-x package-install RET flycheck
```

Настройте pylint с помощью файла pylintrc.

```
pylint --generate-rcfile >.pylintrc
```

Пилинт Odoo плагин

Установите плагин pylint odoo https://github.com/OCA/pylint-odoo

```
pip install --upgrade git+https://github.com/oca/pylint-odoo.git

or

pip install --upgrade --pre pylint-odoo
```

Добавьте плагин в pylintrc.

```
load-plugins=pylint_odoo
```

Полезные конфигурации

По умолчанию в каждой строке допускается 100 символов. Разрешить 120 символов

```
max-line-length=120
```

Чтобы отключить определенное предупреждение, добавьте его код в список “ disable“ в pylintrc. Например, если вам не нравится это сообщение “ Отсутствует метод docstring“ с кодом C0111 или это “ Использование super в классе старого стиля“ (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
```

Вы можете найти другие коды здесь: http://pylint-messages.wikidot.com/

Flychek выделяет строки импорта odoo как “ из моделей импорта openerp, полей, api“ с сообщением об ошибке “ F0401: Невозможно импортировать ... “. Есть два варианта, чтобы исправить это - http://stackoverflow.com/questions/1899436/pylint-unable-to-import-error-how-to-set-pythonpath.

Отредактируйте “ pylintrc“, чтобы включить ваш каталог odoo следующим образом:

```
init-hook='import sys; sys.path.append("/path/to/odoo")'
```

12.2 PyCharm

12.2.1 PyCharm

Удаленный доступ с помощью pgAdmin к базе данных Odoo postgre в Ubuntu

** Это для интеграции с PgAdmin, но такой же метод работает с PyCharm. **

ШАГ № 1 - получить pgAdmin Установить pgAdmin с pgadmin.org

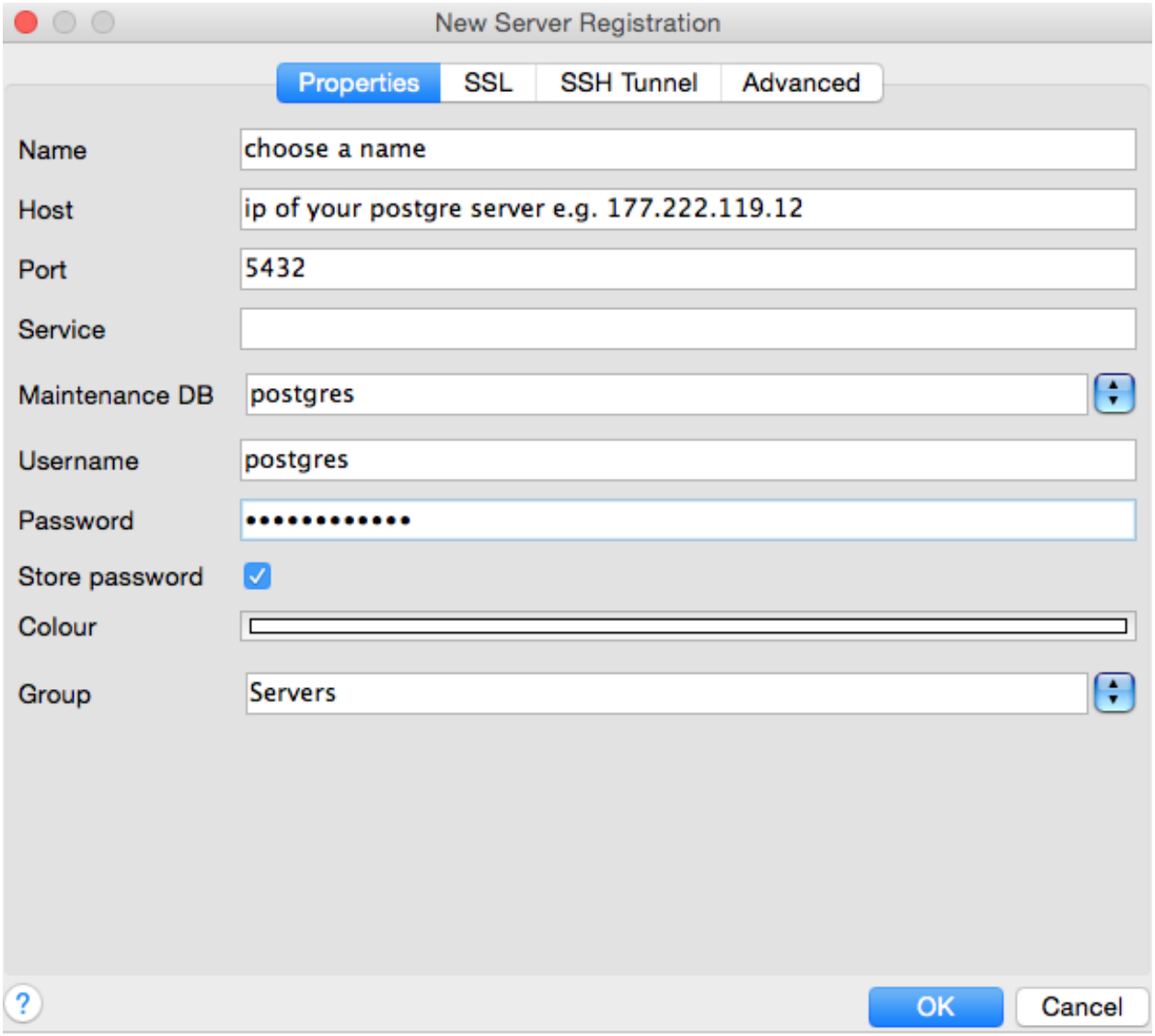
ШАГ # 2 - разрешить удаленное подключение к серверу postgre отовсюду Откройте файл etc / postgresql / 9.x / main / pg_hba.conf и добавьте следующую строку: host all all all md5

ШАГ # 3 - пусть сервер postgre слушает всех. Откройте etc / postgresql / 9.x / main / postgresql.conf и измените следующую строку: listen_addresses = '*'

ШАГ # 4 - дать пользователю «postgres» пароль. Запустите терминал psql: sudo -u postgres psql. Введите пароль: ALTER USER postgres PASSWORD 'yourpassword'; Оставьте терминал psql: q

ШАГ # 5 Перезапустите сервер postgre, выполнив команду терминала: sudo /etc/init.d/postgresql restart

ШАГ # 6 Запустите pgAdmin и добавьте соединение с сервером следующим образом:



Ты готов!

Оригинал:

http://odoo.guide/remote-access-with-pgadmin-to-odoo-postgre-database-on-ubuntu/

12.3 Tmux

12.3.1 Установка Tmux

Установить Tmux

```
sudo apt-get install tmux
```

Проверить версию

```
tmux -V
```

Если у вас есть 1.8 или старше, то вы должны обновить. Вот команды обновления для 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
```

Теперь, если вы выполните “ tmux -V“, он должен показать “ tmux 2.0“, который является хорошей версией для плагинов tmux.

Основано на: http://stackoverflow.com/questions/25940944/ugrade-tmux-from-1-8-to-1-9-on-ubuntu-14-04

Установите Tmux Plugin Manager

Требования: “ tmux“ версия 1.9 (или выше), “ git“, “ bash“

Клон TPM:

```
$ git clone https://github.com/tmux-plugins/tpm ~/.tmux/plugins/tpm
```

Поместите это в конец .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'
```

Перезагрузите среду TMUX, чтобы получить источник TPM:

```
# type this in terminal
$ tmux source ~/.tmux.conf
```

Основано на: https://github.com/tmux-plugins/tpm

Установите Tmux Resurrect

Добавьте плагин в список плагинов TPM в .tmux.conf:

```
set -g @plugin 'tmux-plugins/tmux-resurrect'
```

Нажмите “ prefix + I“, чтобы получить плагин и получить его. Теперь вы должны иметь возможность использовать плагин.

Основано на: https://github.com/tmux-plugins/tmux-resurrect

Установить tmux-континуум

Последнее сохраненное окружение автоматически восстанавливается при запуске tmux. Поместите следующие строки в “ tmux.conf“:

```
set -g @continuum-save-interval '5'
set -g @continuum-restore 'on'
```

Ваша среда будет автоматически сохраняться каждые 5 минут. Когда вы запустите tmux, он автоматически восстановится

Основано на: https://github.com/tmux-plugins/tmux-continuum

12.3.2 Конфигурация Tmux

Создайте файл с именем “ .tmux.conf“ в вашем домашнем каталоге.

Пример “ .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
```

(continues on next page)

(продолжение с предыдущей страницы)

```
# 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'
```

12.4 Visual Studio Code

12.4.1 Установите код Visual Studio

- установить visualstudiocode с https://code.visualstudio.com
- **добавьте следующие расширения:**
  - python: https://marketplace.visualstudio.com/items?itemName=donjayamanne.python
  - odoo-сниппеты: https://marketplace.visualstudio.com/items?itemName=jeffery9.odoo-snippets
- Выполните те же инструкции в (emacs-pylint) для установки pylint и плагина Pylint Odoo. Затем выполните те же настройки в файле pylintrc, как описано здесь.

Внимание: Файл pylintrc может быть помещен в пользовательскую среду для работы со всеми проектами. как для Debian "~ / .pylintrc"

12.4.2 Конфигурация: -

** пример конфигурации (для настройки пользователя или рабочей области) **

```
// 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
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    "files.exclude": {
        "**/*.pyc": true
                        }
}
```

Примечание: некоторые строки закомментированы, потому что это необязательно. Вы можете активировать их при необходимости, как в случае использования Virtualenv.

12.4.3 Отладка

Конфигурации запуска

Для отладки вашего приложения в VS Code вам сначала необходимо настроить файл конфигурации запуска - launch.json. Нажмите на значок «Настроить механизм» в верхней панели окна «Отладка», выберите свою среду отладки, и VS Code сгенерирует файл launch.json в папке .vscode вашего рабочего пространства.

** пример отладки python **

```
{
    "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"
    ]
},
```

Важно: используйте «args» для указания любых опций, таких как база данных, конфигурация или имя пользователя и пароль.

Sorce

Глава 13

Другой

13.1 RST формат

13.1.1 Название документа / Субтитры

Заголовок всего документа отличается от заголовков разделов и может быть отформатирован несколько иначе (например, писатель HTML по умолчанию показывает его в виде центрированного заголовка).

Чтобы указать заголовок документа в reStructuredText, используйте уникальный стиль украшения в начале документа. Чтобы указать субтитры документа, используйте другой уникальный стиль оформления сразу после заголовка документа. Например:

```
================
 Document Title
================
----------
 Subtitle
----------

Section Title
=============

...
```

Обратите внимание, что надписи «Заголовок документа» и «Заголовок раздела» выше используют знаки равенства, но являются отдельными и не связанными стилями. Текст заглавных и подчеркнутых заголовков (но не только подчеркнутых) может быть вставлен для эстетики.

Разделы

- # с надписью, для деталей
- * с подчеркиванием, для глав

- =, для разделов
- -, for subsections
- ^, для подразделов
- ", для пунктов

Блок кода

Введите двойное двоеточие (::) и затем пустую строку, а затем хотя бы один пробел и, наконец, вы можете ввести свой код.

Также вы можете использовать “ ссылку на код“ с помощью “ “ .

13.1.2 выбор

- “ ** жирный ** “
- “ * Курсивный * “
- “ “ *Code*“‘

13.1.3 Списки

- * - не нумеруется
- #. - нумерованный
- 1,2,3, . . . - нумеруются

13.1.4 связи

- внутренняя ссылка

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

- внешняя ссылка:

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

13.1.5 Больше документации

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

13.2 Скрипт настройки размера окна хрома

Вы можете сделать скриншот с нужным вам размером.

Открытый хром. Не расширяйте окно (или в работе с ценами). Запустите эту команду

```
wmctrl -a chromium -e 1,0,0,760,451
```

Последние два аргумента это ширина и высота. Подумайте о том, чтобы добавить рамки окна хрома к размеру вашего скриншота В моем случае это 10px по ширине и 80px по высоте. Скорее всего, вы получили то же самое. Таким образом, для 750 x 371 это будет 760 x 451.

Нужна помощь?

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

- info@itpp.dev
- https://itpp.dev
