# **ODOO模块升级常用脚本及其解释**

本文介绍了一些Odoo模块版本升级的常用技巧和脚本。这些脚本能大大的提高模块升级的效率。 原文请参见：[Module Migration](https://www.it-projects.info/page/module-migration)

## 代码回顾

下列的命令用来评估模块版本迁移大致的工作量，执行命令后可以一次显示所涉模块的所有源码。

# 查看所有源码

find . -iname "\*.py" -or -iname "\*.xml" -or -iname "\*.csv" -or -iname "\*.yml" -or -iname "\*.js" -or -iname "\*.rst" -or -iname "\*.md" | xargs tail -n +1 | less

# 查看所有不包含文档的源码

find . -iname "\*.py" -or -iname "\*.xml" -or -iname "\*.csv" -or -iname "\*.yml" -or -iname "\*.js" | xargs tail -n +1 | less

## 关闭不迁移的模块

# (从上游版本库中创建一个新的分支)

# 将所有模块设置成不可安装（non-installable)

find . -type f -name \_\_openerp\_\_.py -or -name \_\_manifest\_\_.py | xargs sed -i 's/"installable": True/"installable": False/'

find . -type f -name \_\_openerp\_\_.py -or -name \_\_manifest\_\_.py | xargs sed -i "s/'installable': True/'installable': False/"

# 检查没有``installable``标签的模块描述文件

find . -type f -name \_\_openerp\_\_.py -or -name \_\_manifest\_\_.py | xargs grep -L "installable.: "

# (如果发现了这些文件则手动编辑使其installable=False)

# 准备提交

git add .

# 检查diff

git diff --cached

git commit -m "[PORT] mark unported modules as non-installable"

# (推送到上游版本库 "git push" 并在github上请求合并)

## 代码迁移到新的API

### 自动替换

# 所有的Import替换

# 替换 osv, orm

find . -type f -name '\*.py' | xargs sed -i 's/from openerp.osv import orm$/from odoo import models/g'

find . -type f -name '\*.py' | xargs sed -i 's/from openerp.models.orm import Model$/from odoo.models import Model/g' find . -type f -name '\*.py' | xargs sed -i 's/osv.osv\_memory/models.TransientModel/g'

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

find . -type f -name '\*.py' | xargs sed -i 's/osv.except\_osv/UserError/g'

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

find . -type f -name '\*.py' | xargs sed -i 's/\<orm\./models./g'

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

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

find . -type f -name '\*.py' | xargs sed -i 's/\(import .\*\)osv/\1models/g'

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

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

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

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

# 替换 http import

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

find . -type f -name '\*.py' | xargs sed -i 's/openerp.addons.web.http/odoo.http/g'

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

# 替换 odoo

# 导入路径替换。否则会有下面的错误：

# AttributeError: 'module' object has no attribute 'session\_dir'

find . -type f -name '\*.py' | xargs sed -i 's/openerp.tools.config/odoo.tools.config/g'

# 全局替换openerp为odoo

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

# 字段

# 更新字段

# (multiline: http://stackoverflow.com/questions/1251999/how-can-i-replace-a-newline-n-using-sed/7697604#7697604 )

# 删除 `\_columns` 属性

find . -type f -name '\*.py' | xargs perl -i -p0e 's/ \_columns = {(.\*?)\n }/$1\n/gs'

# 计算字段

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

find . -type f -name '\*.py' | xargs sed -i 's/fields.function(\(.\*\) multi=[^,)]\*/fields.function(\1/g'

find . -type f -name '\*.py' | xargs sed -i 's/fields.function(\([^,]\*\)\(.\*\)type=.\([2a-z]\*\)["\x27]/fields.\3(compute="\1"\2/g'

find . -type f -name '\*.py' | xargs sed -i 's/fields.many2one(\(.\*\)obj=\([^,]\*\)/fields.many2one(\2, \1/g'

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

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

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

# 替换 fields

find . -type f -name '\*.py' | xargs perl -i -p0e 's/ \_columns = {(.\*?) }/$1/gs'

find . -type f -name '\*.py' | xargs sed -i 's/fields\.\(.\)/fields.\u\1/g'

find . -type f -name '\*.py' | xargs sed -i 's/ [\x27"]\(.\*\)[\x27"].\*:.\*\(fields.\*\),$/\1 = \2/g'

# 更名 attributes

find . -type f -name '\*.py' | xargs sed -i 's/select=/index=/g'

find . -type f -name '\*.py' | xargs sed -i 's/digits\_compute=/digits=/g'

### 半自动替换

我们推荐在提交了上述修改之后，使用下面的命令来检查并手工做更新。

# pool -> env

find . -type f -name '\*.py' | xargs sed -i 's/self.pool/self.env/g'

# 去除 cr, uid

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

find . -type f -name '\*.py' | xargs sed -i 's/(self, cr, [^,]\*, ids/(self/g'

find . -type f -name '\*.py' | xargs sed -i 's/(self, cr, uid, /(self, /g'

find . -type f -name '\*.py' | xargs sed -i 's/, context=[^,)]\*//g'

find . -type f -name '\*.py' | xargs sed -i 's/self.env.get(\([^)]\*\))/self.env[\1]/g'

# res\_config.py

find . -type f -name 'res\_config.py' | xargs sed -i 's/\(def get\_default\_.\*\)(self)/\1(self, fields)/g'

### 10.0+ 更新

# 将``\_\_openerp\_\_.py``更名为``\_\_manifest\_\_.py``

find . -type f -name \_\_openerp\_\_.py -exec rename 's/\_\_openerp\_\_.py/\_\_manifest\_\_.py/' '{}' \;

### 11.0+ 更新

# ir.config\_parameter -- 在``get\_param``/``set\_param``前添加``sudo()``

find . -type f -name '\*.py' | xargs perl -i -p0e 's/(?<!sudo\(\)\.)(get\_param|set\_param)/sudo().$1/g'

## 引用名的迁移

### 9.0- → 10.0+

# menu\_hr\_configuration

find . -type f -name '\*.xml' | xargs sed -i 's/menu\_hr\_configuration/menu\_human\_resources\_configuration/g'

# base.group\_hr

find . -type f -name '\*.csv' -o -name '\*.py' -o -name '\*.xml' | xargs sed -i 's/base.group\_hr/hr.group\_hr/g'

# website.salesteam\_website\_sales

find . -type f -name '\*.csv' -o -name '\*.py' -o -name '\*.xml' | xargs sed -i 's/website.salesteam\_website\_sales/sales\_team.salesteam\_website\_sales/g'

# base.group\_sale\_salesman

find . -type f -name '\*.csv' -o -name '\*.py' -o -name '\*.xml' | xargs sed -i 's/base.group\_sale\_salesman/sales\_team.group\_sale\_salesman/g'

# product.prod\_config\_main

find . -type f -name '\*.xml' | xargs sed -i 's/product.prod\_config\_main/sale.prod\_config\_main/g'

### 10.0- → 11.0+

# mixins in js

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

# 11.0 没有 website.config.settings

find . -type f -name '\*.py' -o -iname '\*.xml' | xargs sed -i 's/website\.config\.settings/res.config.settings/g'

# page="True" 不再使用了

find . -type f -name '\*.xml' | xargs sed -i 's/ page="True"//g'

## 迁移到python3

# urlparse

find . -type f -name '\*.py' | xargs sed -i 's/import urlparse/import urllib.parse as urlparse/g'

find . -type f -name '\*.py' | xargs sed -i 's/from urlparse/from urllib.parse/g'

# Odoo cmdline批量升级已安装模块

2018年02月26日

cmd命令为:

sudo /opt/odoo/odoo-bin *--addons-path=addons,XXA\_addons,XXB\_addons --db\_user=odoo --db\_password=odoo*

*--db\_host=127.0.0.1 -u module\_A,module\_B --db-filter youraccountname -d youraccountname*

XXA\_addons:你的addons

XXB\_addons:其他人的addons

module\_A: 已安装的模块名称

# youraccountname:账套名称

# **用命令单独升级模块**

很多时候，我们在二开模块之后，需要再次调整需求，这可能会对原有模块中的代码有较大的功能调整，如果说直接将模块放入代码库——重启——升级，则有可能会造成整个数据库的崩溃，那么该如何解决该问题呢？

如果将模块代码更新完成，已经知道对原有模块调整较大，我们可以不采用传统的方式升级模块。直接通过命令单独升级需要的模块即可。

**方法如下**

用命令单独升级模块

**--**config**=**G:\project\odoo**-**10.0\openerp**-**server**.**conf **-**u module **-**d database

代码解释：

**--**config**=**G:\project\odoo**-**10.0\openerp**-**server**.**conf

是配置文件的路径

module

待升级的模块名

database

待升级的数据库名