**OODO设计模式一消息通知**

## **问题**

Odoo系统根据单据更新发送消息通知相关人员。

举例：  
生产订单下发，库管人员就会得到需要发货的消息。

相关的设计模式有：

* 重写动作函数.
* 自动化动作
* 定制开发，根据单据状态，发送消息

下面就，讨论不同模式的实现方法，和利弊。

## **重写动作函数**

odoo v8 中 确认生产订单 通过workflow完成，  
workflow 调用的确认函数是:

def action\_confirm(self, cr, uid, ids, context=None):

""" Confirms production order.

@return: Newly generated Shipment Id.

"""

user\_lang = self.pool.get('res.users').browse(cr, uid, [uid]).partner\_id.lang

context = dict(context, lang=user\_lang)

uncompute\_ids = filter(lambda x: x, [not x.product\_lines and x.id or False for x in self.browse(cr, uid, ids, context=context)])

self.action\_compute(cr, uid, uncompute\_ids, context=context)

for production in self.browse(cr, uid, ids, context=context):

self.\_make\_production\_produce\_line(cr, uid, production, context=context)

stock\_moves = []

for line in production.product\_lines:

if line.product\_id.type != 'service':

stock\_move\_id = self.\_make\_production\_consume\_line(cr, uid, line, context=context)

stock\_moves.append(stock\_move\_id)

else:

self.\_make\_service\_procurement(cr, uid, line, context=context)

if stock\_moves:

self.pool.get('stock.move').action\_confirm(cr, uid, stock\_moves, context=context)

production.write({'state': 'confirmed'})

return 0

重写函数, 增加以下代码实现。

SUPER(mrp\_prodduction, self).action\_confirm(cr, uid, ids, context=None)

self.message\_post(cr, uid, ids, body=\_("Order %s confirmed. Please Send Material") % self.\_description, context=context)

修改结果



rewrite\_function.png

优点: 简单直接  
缺点: 需要找到代码函数，重写函数。

## **定制，根据状态变化**

此方法需要修改代码，在单据状态变化的时候，自动推送消息。

依赖代码部分  
模块集成

\_inherit = ['mail.thread', 'ir.needaction\_mixin']

定义状态, 增加track\_visibility属性

'state': fields.selection(

[('draft', 'New'), ('cancel', 'Cancelled'), ('confirmed', 'Awaiting Raw Materials'),

('ready', 'Ready to Produce'), ('in\_production', 'Production Started'), ('done', 'Done')],

string='Status', readonly=True,

track\_visibility='onchange', copy=False,

定义\_trace 字段以及参数  
ps. mrp.production 中未定义\_track, 故状态更新 不会推送消息通知。

# Automatic logging system if mail installed

# \_track = {

# 'field': {

# 'module.subtype\_xml': lambda self, cr, uid, obj, context=None: obj[state] == done,

# 'module.subtype\_xml2': lambda self, cr, uid, obj, context=None: obj[state] != done,

# },

# 'field2': {

# ...

# },

# }

# where

# :param string field: field name

# :param module.subtype\_xml: xml\_id of a mail.message.subtype (i.e. mail.mt\_comment)

# :param obj: is a browse\_record

# :param function lambda: returns whether the tracking should record using this subtype

其中 module.subtype\_xml 需要在xml中定义消息类型。 例如 account\_voucher 的跟踪消息类型

<!-- Voucher-related subtypes for messaging / Chatter -->

<record id="mt\_voucher\_state\_change" model="mail.message.subtype">

<field name="name">Status Change</field>

<field name="res\_model">account.voucher</field>

<field name="default" eval="False"/>

<field name="description">Status changed</field>

</record>

优点：根据状态或其他字段自动推送消息。  
缺点：定义复杂。

## **自动化动作**

创建自动话动作，定义对象和条件



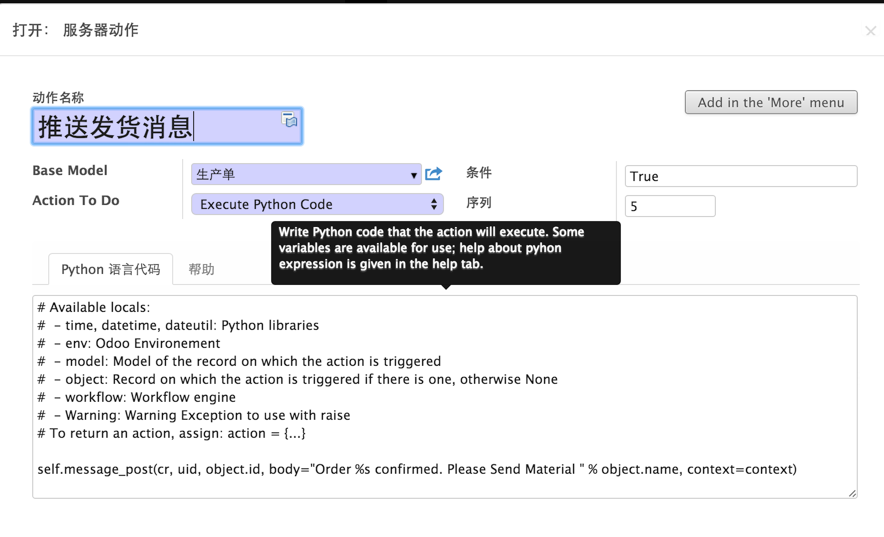
Automatci\_action.png

定义动作: 更改负责人 或增加关注者(本例中可以增加仓库人员)



set\_action1.png

或 更复杂动作，用服务器动作定义



create\_server\_action.png

优点 : 用户可配置  
缺点: server action 需要写python代码

# **总结**

以上三种方法，都是使用message\_post方法发送消息给关注者，如需使用其他发送消息方法，需要在mail thread寻找新的方法。  
方法三，可以自定义配置条件，也可以增加关注者，也可以增加复杂动作，灵活。  
方法一，对开发者来说更直接。