# odoo中根据模型生成viwe视图的框架代码

2016年07月26日

工作一段时间了，由于后台是odoo，而odoo得view视图根据model中定义的类型进行生成，由于每次都要重复写很多东西，而且还容易出错，因此我写了一个根据model生成view视图的框架，生成好后，根据模型中定义的fields再填充数据进去，提高了一点效率。

#coding=utf-8

import re

import os

tmp\_head\_view = """<?xml version="1.0" encoding="utf-8"?>

<openerp>

<data>"""

tmp\_form\_head = """ <record id="{}" model="ir.ui.view">

<field name="name">{}</field>

<field name="model">{}</field>

<field name="arch" type="xml">

<form string="{}">

<group>"""

tmp\_field\_view =""" <field name="{}" />"""

tmp\_form\_last=""" </group>

</form>

</field>

</record>"""

tmp\_tree\_head = """ <record id="{}" model="ir.ui.view">

<field name="name">{}</field>

<field name="model">{}</field>

<field name="type">tree</field>

<field name="arch" type="xml">

<tree string="{}" >"""

tmp\_tree\_last = """ </tree>

</field>

</record>"""

tmp\_search\_head = """ <record id="{}" model="ir.ui.view">

<field name="name">{}</field>

<field name="model">{}</field>

<field name="arch" type="xml">

<search string="Search Payslips">"""

tmp\_search\_last = """ </search>

</field>

</record>"""

tmp\_act\_view = """ <record id="{}" model="ir.actions.act\_window">

<field name="name">{}</field>

<field name="res\_model">{}</field>

<field name="view\_type">form</field>

<field name="view\_mode">tree,form</field>

<field name="view\_id" ref="{}"/>

<field name="search\_view\_id" ref="{}"/>

</record>"""

tmp\_menu\_view = """ <menuitem id="{}" name="{}" sequence="10" action="{}" parent="{}"/>"""

tmp\_last\_view = """ </data>

</openerp>"""

name\_re = re.compile(r"\'([^\']\*)\'")

name\_line\_re = re.compile(r" \_name")

field\_re = re.compile(r"fields\.")

def write\_view\_from\_file(file\_name,target\_file):

fwrite = open(target\_file, 'w')

fwrite.write(tmp\_head\_view)

fwrite.write("\n")

fwrite.close()

if os.path.exists(file\_name):

if os.path.isfile(file\_name):

write\_view\_config(file\_name,target\_file)

def write\_view\_config(file\_name,target\_file):

fwrite = open(target\_file, 'a')

with open(file\_name, 'r') as f:

line = 0

line\_list = []

for line in f.readlines():

if name\_line\_re.findall(line):

print(line)

names = name\_re.findall(line.strip())

name = names[0]

name\_normal = name.replace('.','\_')

name\_form\_id = name\_normal+"\_form"

name\_tree\_id = name\_normal+"\_tree"

name\_filter\_id = name\_normal+"\_filter"

name\_act\_id = "act\_"+name\_normal

name\_menu\_id = "menu\_"+name\_normal

fwrite.write(tmp\_form\_head.format(name\_form\_id,name\_form\_id,name,name\_form\_id))

fwrite.write("\n")

fwrite.write(tmp\_form\_last)

fwrite.write("\n")

fwrite.write("\n")

fwrite.write(tmp\_tree\_head.format(name\_tree\_id,name\_tree\_id,name,name\_tree\_id))

fwrite.write("\n")

fwrite.write(tmp\_tree\_last)

fwrite.write("\n")

fwrite.write("\n")

fwrite.write(tmp\_search\_head.format(name\_filter\_id,name\_filter\_id,name))

fwrite.write("\n")

fwrite.write(tmp\_search\_last)

fwrite.write("\n")

fwrite.write("\n")

fwrite.write(tmp\_act\_view.format(name\_act\_id," ",name,name\_tree\_id,name\_filter\_id))

fwrite.write("\n")

fwrite.write("\n")

fwrite.write(tmp\_menu\_view.format(name\_menu\_id," ",name\_act\_id," "))

fwrite.write("\n")

fwrite.write("\n")

fwrite.write(tmp\_last\_view)

fwrite.close()

fwrite.close()

if \_\_name\_\_ =='\_\_main\_\_':

file\_names = [

'model\_config.py' #写入模型文件名,可以绝对路径和相对路径

]

for file\_name in file\_names:

if file\_name.find(".py")>0:

new\_file\_name = file\_name.split(".")[0]+"\_view\_tmp.xml"

write\_view\_from\_file(file\_name,new\_file\_name)

else:

print("file: {} is not a valid file".format(file\_name))