# 简单的请假模块

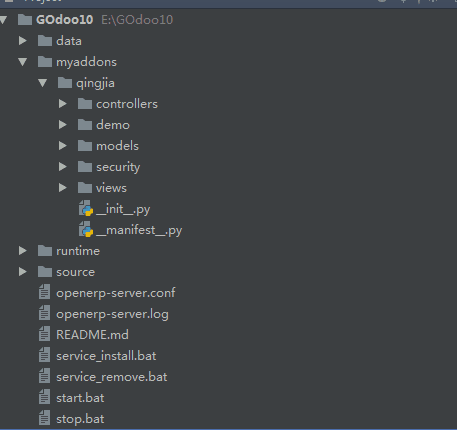
在windows cmd命令行下：

e:

cd E:\GOdoo10

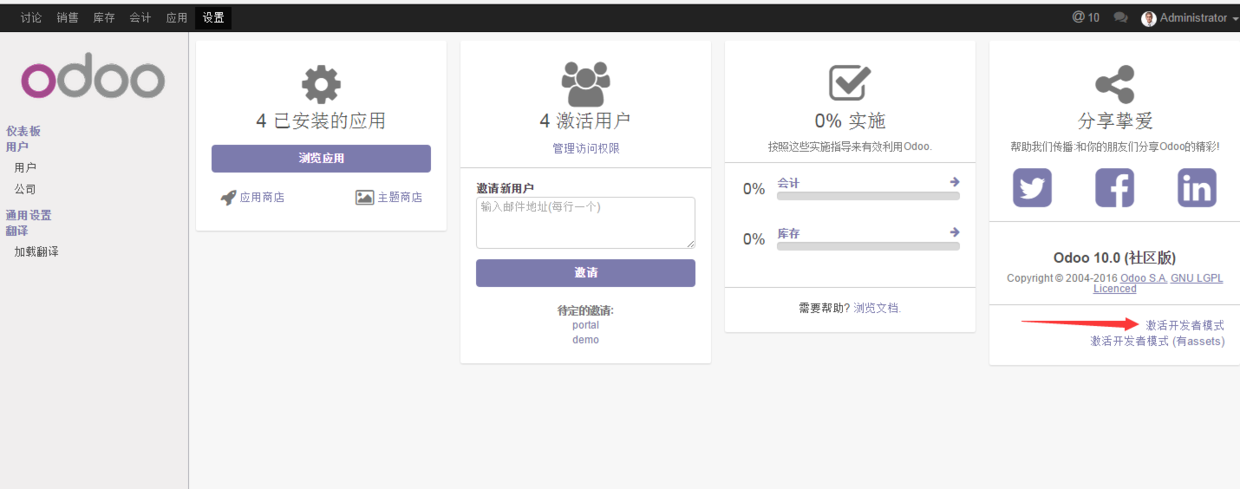
E:\GOdoo10\runtime\python\python source\odoo-bin scaffold qingjia myaddons

E:\GOdoo10 是odoo10绿色版的解压目录。  
E:\GOdoo10\runtime\python\python 这个是绿色版自带的python环境，source\odoo-bin 是odoo10源码中的odoo命令入口，scaffold 是odoo的脚手架功能，可以创建一个标准的odoo模块框架，qingjia是创建的模块名，myaddons是模块所放置的目录路径。模块加载目录有两个，一个是GOdoo10\myaddons，另一个是GOdoo10\source\addons，但是按约定GOdoo10\source\addons放系统模块，GOdoo10\myaddons放自己开发的模块。



请假单目录结构

建立完成后，目录是这样的，这里是用pycharm进行开发的，也可以用vs2015进行开发。



激活开发者模式

以管理员身份登录系统，设置>激活开发者模式，odoo中很多操作和设置是需要管理员身份在开发者模式下才能进行的。



安装请假模块

应用>更新应用列表，更新应用列表是为了能够搜索到刚刚添加的qingjia模块。更新完成后，在搜索栏关闭“应用”标签，然后输入qingjia搜索，能看到我们的模块了。安装qingjia模块，安装完成后会发现什么也没增加，正常，因为脚手架中还没有加入任何代码。现在开始加入代码，编辑文件\_\_manifest\_\_.py

# -\*- coding: utf-8 -\*-

{

'name': "qingjia",

'summary': """

请假模块""",

'description': """

请假模块

""",

'author': "leo",

'website': "http://www.yourcompany.com",

# Categories can be used to filter modules in modules listing

# Check https://github.com/odoo/odoo/blob/master/openerp/addons/base/module/module\_data.xml

# for the full list

'category': 'Uncategorized',

'version': '0.1',

# any module necessary for this one to work correctly

'depends': ['base'],

# always loaded

'data': [

# 'security/ir.model.access.csv',

'views/views.xml',

'views/templates.xml',

],

# only loaded in demonstration mode

'demo': [

'demo/demo.xml',

],

}

这是整个模块的配置文件，配置项从字面上很容易理解。  
name：模块名，会显示在模块列表中。  
summary：摘要，在模块详情显示。  
description：描述，在模块详情显示。  
depends：模块依赖项，在这个模块中没有需要依赖的其它模块，保留默认的base。  
data：模块资源文件，视图、菜单、权限等都可以写在资源文件中。  
demo：模块演示数据。

修改模块的类文件 models/models.py

from odoo import models, fields, api

class qingjiadan(models.Model):

\_name = 'qingjia.qingjiadan'

name = fields.Char(string="申请人")

days = fields.Integer(string="天数")

startdate = fields.Date(string="开始日期")

reason = fields.Text(string="请假事由")

这里我们定义了qingjiadan对象模型，包含四个属性，name，days，startdate，reason。在模块安装完成后，odoo的ORM框架会自动把这个对象映射到数据库表。属性类型会映射到表字段数据类型，表名是模块名\_对象名，比如这个对象对应的表名是qingjia\_qingjiadan

修改资源文件 views/views.xml

<openerp>

<data>

<!-- tree视图 -->

<record id="view\_tree\_qingjia\_qingjiadan" model="ir.ui.view">

<field name="name">请假单列表</field>

<field name="model">qingjia.qingjiadan</field>

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

<tree>

<field name="name"/>

<field name="days"/>

<field name="startdate"/>

</tree>

</field>

</record>

<!-- form视图 -->

<record id="view\_form\_qingjia\_qingjiadan" model="ir.ui.view">

<field name="name">请假单</field>

<field name="model">qingjia.qingjiadan</field>

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

<form>

<sheet>

<group name="group\_top" string="请假单">

<field name="name"/>

<field name="days"/>

<field name="startdate"/>

<field name="reason"/>

</group>

</sheet>

</form>

</field>

</record>

<!-- 视图动作 -->

<act\_window id="action\_qingjia\_qingjiadan"

name="请假单"

res\_model="qingjia.qingjiadan"

view\_mode="tree,form" />

<!-- 顶级菜单 -->

<menuitem name="请假" id="menu\_qingjia"/>

<!-- 二级菜单 -->

<menuitem name="请假单" id="menu\_qingjia\_qingjiadan" parent="menu\_qingjia" action="action\_qingjia\_qingjiadan"/>

</data>

</openerp>

这里定义了一个tree视图，一个form视图，一个视图动作，还有两个菜单。  
tree视图用于显示请假单列表页面。  
<record id="view\_tree\_qingjia\_qingjiadan" model="ir.ui.view">  
id tree视图的全局唯一标识  
model 资源类型，tree视图和form视图都是ir.ui.view，这里对应ir\_ui\_view数据库表，模块安装后，资源数据会写入对应的数据库表中。  
<field name="model">qingjia.qingjiadan</field>  
将这个视图与我们之前定义的对象模型qingjia.qingjiadan进行绑定。

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

<tree>

<field name="name"/>

<field name="days"/>

<field name="startdate"/>

</tree>

</field>

这里表示这是一个tree视图，并定义列表项显示的列。在列表项中显示name,days,startdate三个字段的内容，这里字段都是在qingjia.qingjiadan对象模型中定义的。  
form视图用于显示请假单详情页，定义方式与tree视图类似。有两个特殊的容器<sheet><group>是用于页面布局的。

<act\_window id="action\_qingjia\_qingjiadan"

name="请假单"

res\_model="qingjia.qingjiadan"

view\_mode="tree,form" />

这里定义视图动作，视图动作将菜单、视图、模型进行关联。  
name 会在模块的导航条中显示  
res\_model 视图动作绑定的模型  
view\_mode 视图动作关联的视图类型

<!-- 顶级菜单 -->

<menuitem name="请假" id="menu\_qingjia"/>

<!-- 二级菜单 -->

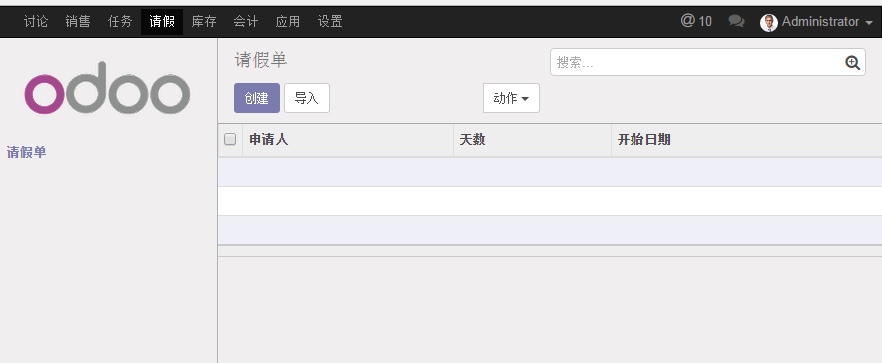
<menuitem name="请假单" id="menu\_qingjia\_qingjiadan" parent="menu\_qingjia" action="action\_qingjia\_qingjiadan"/>

这里定义了两级菜单，顶级菜单将出现在odoo导航菜单上，二级菜单的通过parent属性与顶级菜单关联，action是菜单点击动作响应方法。



模块升级

修改完代码后，重启odoo服务，重新登录系统，在应用>应用中再次找到我们之前安装过的qingjia模块，进入模块详情，把qingjia模块升级。



请假单tree视图



请假单form视图

好了，现在可以看到请假模块界面，能实现基本的增删改查功能。

# 数据验证

在很多场景下，需要对用户输入数据进行验证。比如编码不能重复，邮箱的准确性，手机号码规则，凡是不符合这些验证规则的数据，要报错并拒绝保存。在odoo中使用constraints（约束）来实现。odoo提供两种方式实现自动验证，python constraints和sql constraints

sql constraints

\_sql\_constraints = [  
('name\_description\_check',  
'CHECK(name != description)',  
"名称和描述不能相同"),

('name\_unique',

'UNIQUE(name)',

"已存在相同名称"),

]  
sql constraints 通过模块属性\_sql\_constraints 进行定义。它是一个元组的列表，每个元组是一条数据约束(name, sql\_definition, message)，包含三个字符串元素：1）约束名称；2）约束规则（postgresql约束规则）；3）违反约束规则时的警告信息。

注意,使用sql constraints，需要确保当前数据库里面没有违反该约束的数据，如果数据库中存在有违反约束的记录，那么在更新模块的时候系统日志里面会有警告信息并且constraints会添加失败。  
python constraints

from odoo.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

@api.constrains('instructor\_id', 'attendee\_ids')  
def \_check\_instructor\_not\_in\_attendees(self):  
for r in self:  
if r.instructor\_id and r.instructor\_id in r.attendee\_ids:  
raise exceptions.ValidationError("A session's instructor can't be an attendee")  
python constraints，是通过装饰器@api.constrains(field\_name,)来定义，每次记录修改的时候，如果包含装饰器定义的字段就会触发下面的方法，所以需要在方法里面判断是否违反约束，如果违反，则通过raise异常来弹出警告框并阻止记录保存。使用python constraints的时候就算是系统内已经有违反约束的记录也可以对新记录生效，并且里面的判断可以添加更加复杂的逻辑。

总的来说，sql constraints的效率较高，而python constraints使用更加灵活。