openerp/modules/loading.py

load\_modules方法

模块载入分为如下9个阶段，

1. 加载base类中的模块
2. 标记其他模块为已加载或者已升级
3. 加载已标记模块，这个步骤分为两个子阶段，1：加载标记为installed或者to upgrade或者to remove的模块，但是to remove模块会在第6步移除。2：安装新模块
4. 完成安装后续步骤
5. 清除无效菜单
6. 卸载待移除模块
7. 检查自定义视图
8. 调用所有模块的\_register\_hook方法
9. 运行模块测试

ir\_ui\_view.render 中id\_or\_xml\_id参数格式为aaa.bbb 其中aaa代表模块名，bbb代表数据库ir\_model\_data中name字段的值，对应行中的res\_id又指向ir\_ui\_view表中的id。

读取模板是在ir\_ui\_view.py中read\_template方法

他会调用ir.model.data类中的xmlid\_to\_res\_id方法读取，ir.model.data的读取方法会先以点号作为分隔符，第一部分是模块名，第二部分对应数据库中的name字段。找到指定数据后，res\_id字段表明ir\_ui\_view表中的id，再查到那条数据，那条数据就是xml模板视图数据。

静态资源的跳转在http.py **load\_addons（）**方法中

继承：

在类BaseModel里

create\_instance

**主线**

**server.py -> preload\_registryes -> RegistryManager.new() -> loading.py ->load\_modules() -> registry初始化 -> load\_module\_graph() -> 遍历graph得到index，package -> registry.load() -> 遍历根据package名字取到的openerp.osv.orm.MetaModel.module\_to\_models中的值cls -> cls.create\_instance**

渲染主线：

request.render -> response.render -> ir\_ui\_view.render -> ir\_qweb.render -> render\_node

## BaseModel的create\_instance方法

**注意：以下代码基于odoo四月份的trunk版，新版本中create\_instance方法已经被\_build\_model方法替换，但是原理基本一样。**

如果没有耐心看原理的，可以直接看结论：

1. 本文主要分析odoo模型对象初始化过程和继承过程。
2. odoo中对象不是直接创建，而是调用**类函数create\_instance创建**，并且由于在\_\_new\_\_方法中返回的是None，所以即使**我们使用小括号创建对象也不会返回新对象。**
3. odoo可以多继承，当”\_inherit”字段为列表时，就使用了odoo的对象多继承，如果是字符串，则只有一个父对象，但是并**不建议使用多继承**。
4. 如果使用对象多继承的时候，**同名的方法和属性都是以第一个出现的父类中的方法和属性为准，继承后面不会覆盖前面。**举个例子，基类BaseModel中的create方法不会被其他父类的create方法覆盖。

我们知道，odoo中自己实现的模型都是继承自BaseModel这个基类，同时BaseModel也实现了一系列模型的基本操作，如search、create、write等方法，我们在操作模型的时候可以直接调用父类的相应方法。create\_instance就是BaseModel众多方法其中之一，但是这个方法非常特殊。首先这是一个类方法，如下图：

C:\Users\xiaoyang123\Desktop\QQ截图20140901163713.png

这意味着调用它的是类而不是对象实例。

首先该方法会判断对象是否有父类，即\_inherit字段是否存在，（注意，里的父类是指odoo层面的父类，而不是python级别的父类），如果有父类，则会遍历所有父类（因为有可能是继承多个父类，如果你\_inherit字段是列表类型而不是字符串时，就是多继承），遍历过程都会进行如下操作：

1、记录父类的\_\_class\_\_属性：

C:\Users\xiaoyang123\Desktop\QQ截图20140901175913.png

2、遍历如下列表，并且copy给子类：

C:\Users\xiaoyang123\Desktop\QQ截图20140901172558.png

遍历如下图所示，先定义一个空字典nattr，对每个属性都从父类中copy一份：

C:\Users\xiaoyang123\Desktop\QQ截图20140902141629.png

处理遍历时会对”\_columns”、”\_constraints”进行特殊处理，如果其他属性是字典类型，则直接把原属性更新到new中，如下图所示：

C:\Users\xiaoyang123\Desktop\QQ截图20140902144124.png

对\_columns是在上面那行代码之前进行处理的，因为\_columns也是字典类型，相当于对这个字段进行预处理之后再进行常规处理。如果\_columns中的字段具有manual属性，则不继承该字段。

如果是列表属性，则直接扩展，但是attributes里面处理\_constraints是list类型外其他的都是字典类型，而\_constraints已经被特殊处理了，也就是说下面这行代码正常情况下是不会被调用到的。

C:\Users\xiaoyang123\Desktop\QQ截图20140902144223.png

最后生成一个新的type，如下图：

C:\Users\xiaoyang123\Desktop\QQ截图20140901175506.png

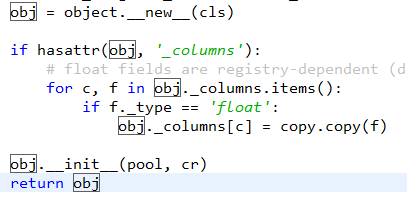
需要注意的是，虽然type的第三个参数dict中的nattr会覆盖同名的属性，但是nattr中的同名属性已经在前面的代码中被子类的同名属性覆盖了一次，所以不会造成影响。

C:\Users\xiaoyang123\Desktop\QQ截图20140902144124.png

此时这个cls已经具备了该父类的方法定义和属性了。然后不断重复上述动作，直到遍历完所有父类。遍历完所有父类之后，cls就具备了所有父类的方法。

需要注意的是由于cls本身是BaseModel，所以，当父类覆写了BaseModel方法的时候，继承对象并不会继承父类的覆写方法，只会继承父类新增的方法，同理，如果两个父类有相同的方法，子类会继承第一个父类的方法（继承方法调用基于先根序）。在我们实际编写模块时需要注意这一点。

当需要创建的对象的类型构建完成之后，程序会调用object的\_\_new\_\_方法创建对象，然后手动调用对象的\_\_init\_\_方法进行初始化操作，如下图：



在\_\_init\_\_方法中会把自己添加到registry中去，所以虽然该方法是由registry的load方法调用，但是对象注册到registry是由对象自身完成的。