

# ODOO / OPENERP

---

PYTHON 环境的快速数据库应用开发

李维

[liwei@sandwych.com](mailto:liwei@sandwych.com)

# 内容概要

---

传统管理系统开发的痛点

Odoo 简介

实际开发演示

总结

# 传统技术的痛点

---

# 定义：以数据库为中心的业务应用

---

各种管理系统：

- 进销存
- 物流管理系统
- 人事管理系统
- XXX 政务系统
- 各种行业垂直系统

# 静态语言（Java & C#）很费劲

---

大量样板代码用于链接语言和外部世界（想想 Java 的 Hibernate）

缺乏动态语言的灵活性

需要编译运行，开发效率低

C# 支持 dynamic，但静态动态代码混合非常别扭

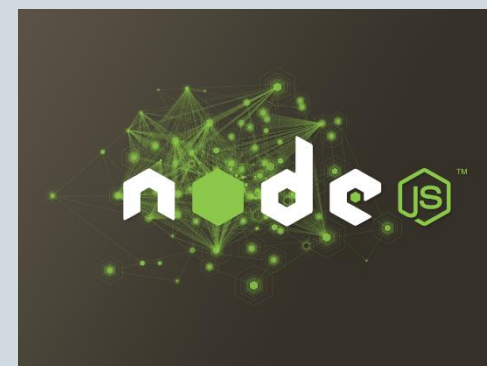
代码性能很高，但管理系统的瓶颈在 IO（数据库、网络连接 etc.）



# 顺带黑一下 NODE.JS

---

管理系统往往业务逻辑复杂，但吞吐量是可以预估的  
数据库采用异步操作将大大增加业务逻辑实现的复杂性





---

原来叫 OpenERP，8.0 以后改名叫 Odoo： <http://www.odoo.com>

以 GPL3 开源的跨平台 Python 业务应用框架及ERP系统

纯 HTML5 界面

对第三方友好：方法及数据均可通过 JSON-RPC/XML-RPC 访问

完全模块的结构：

- 可利用现有模块做二次开发
- 也可只用基础框架完全开发非 ERP 系统

# ODOO 的“三层”架构

---





# 简单的销售订单管理

---

功能：

- 产品信息 CRUD
- 客户信息 CRUD
- 销售订单 CRUD

模块名称：sales

# 模型（表）定义分析

---

产品信息

sales.product

客户信息

sales.customer

销售单主表

sales.order

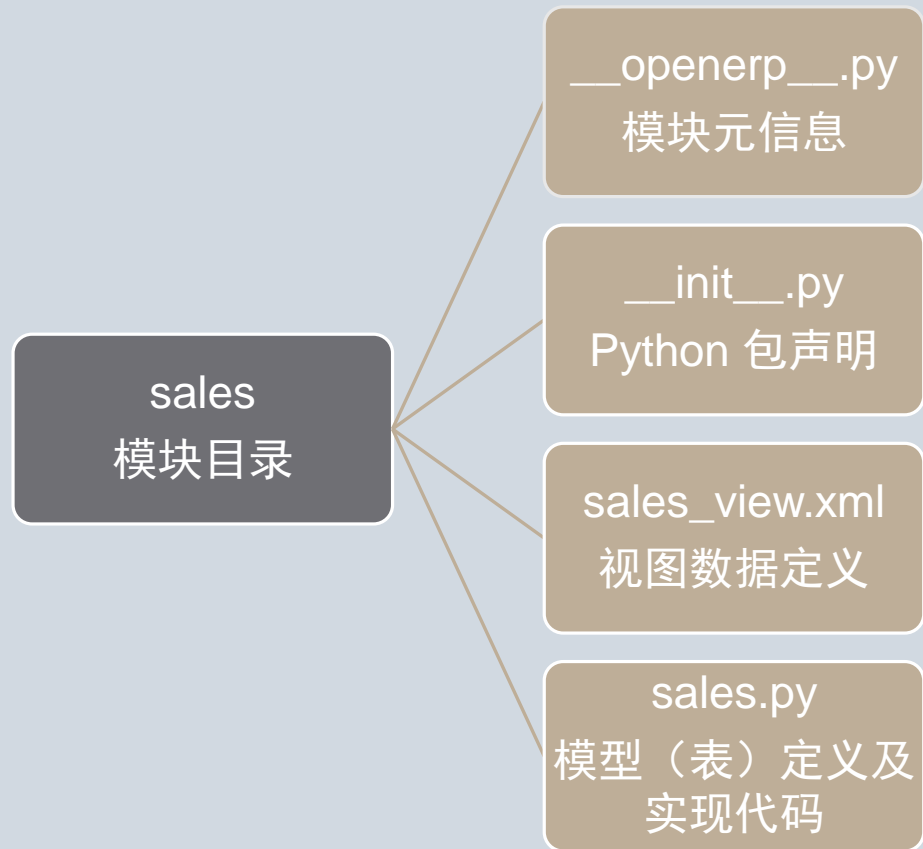
销售单明细表

sales.order.line



# 模块结构

---



# 模块元信息文件：\_\_openerp\_\_.py

---

```
1 #encoding: utf-8
2
3 {
4     'name': u'销售订单管理开发实例', #模块名称, 必填
5     'version': '0.1', #版本
6     'depends': ['base', 'web'], #依赖的模块
7     'category': 'Demo', #模块分类
8     'summary': 'Odoo 简单模块开发例子：销售订单管理', #模块简介
9     'description': """", #模块描述
10    'author': 'YourName', #作者
11    'website': 'http://www.sandwych.com', #
12    'data': [
13        'sales_view.xml', #初始化模块或者升级模块时导入的数据
14    ],
15    'demo': [], #这里指定演示数据
16    'installable': True, #模块是否可通过管理界面安装
17    'images': [], #指定模块的图标等
18 }
```

# 标准的包导入文件：\_\_init\_\_.py

---

```
1 #encoding: utf-8  
2  
3 import sales
```

# 简单的 ODOO 模型 (sales.py)

```
1 # encoding: utf-8
2
3 from datetime import timedelta
4 from datetime import date, datetime
5
6 from openerp import models, fields, api, _
7 from openerp.tools import cache
8 from openerp.exceptions import Warning
9 from openerp.tools import DEFAULT_SERVER_DATE_FORMAT, DEFAULT_SERVER_DATETIME_FORMAT
10
11 class Product(models.Model):
12     u'''产品''' #可选的描述文本, docstring 格式
13     _name = 'sales.product' #定义模型的内部名称, 必须具备
14
15     name = fields.Char(u'名称', required=True, index=True) #名称字段定义
16     unit_price = fields.Float(u'单价', required=True) #单价字段定义
17
```

无聊的标准模块导入,  
blah...blah...

# 视图和菜单： sales\_view.xml

```
<record id="view_sales_product_tree" model="ir.ui.view">
  <field name="name">sales.product.tree</field>
  <field name="model">sales.product</field>
  <field name="arch" type="xml">
    <tree string="产品">
      <field name="name" />
      <field name="unit_price"/>
    </tree>
  </field>
</record>
<record id="view_sales_product_form" model="ir.ui.view">
  <field name="name">sales.product.form</field>
  <field name="model">sales.product</field>
  <field name="arch" type="xml">
    <form string="产品">
      <sheet>
        <group>
          <field name="name" />
          <field name="unit_price"/>
        </group>
      </sheet>
    </form>
  </field>
</record>
<record id="action_sales_product" model="ir.actions.act_window">
  <field name="name">产品</field>
  <field name="res_model">sales.product</field>
  <field name="view_type">form</field>
  <field name="view_mode">tree,form</field>
</record>
<menuitem name="产品管理" action="action_sales_product" id="menu_sales_product" parent="menu_sales_config" sequence="10" />
```

产品列表视图定义

产品表单视图定义

菜单动作定义

菜单项定义

# 关于视图的一些解释

---

视图是通过 XML 定义的数据导入数据库中，Odoo 中的一切都是保存在数据库里的

Odoo 支持多种视图：树形/表单/日历/看板/图表，也支持使用 HTML+CSS+JS 自定义

没错，列表也是用 tree 来定义的，Odoo 允许显示层次数据

表单视图包含多种部件（widget），如：

- 多对一字段可使用默认 widget 和标准下拉列表两种方式显示
- 浮点数字段可使用数字文本框、进度条、时间框多种方式显示



# 总结一下我们刚才干了些什么

---

1. 定义了一个简单的 Odoo 模块
2. 创建了一个产品的模型
3. 定义了产品的视图、菜单动作及菜单项

# 安装我们开发的模块

Local Modules / 销售订单管理开发实例

更多 ▾



## 销售订单管理开发实例

By YourName

安装

信息

技术数据

网站	<a href="http://www.sandwych.com">http://www.sandwych.com</a>	技术名称	sales
分类	Demo	许可	Affero GPL-3
摘要	Odoo 简单模块开发例子：销售订单管理	最新版本	8.0.0.1

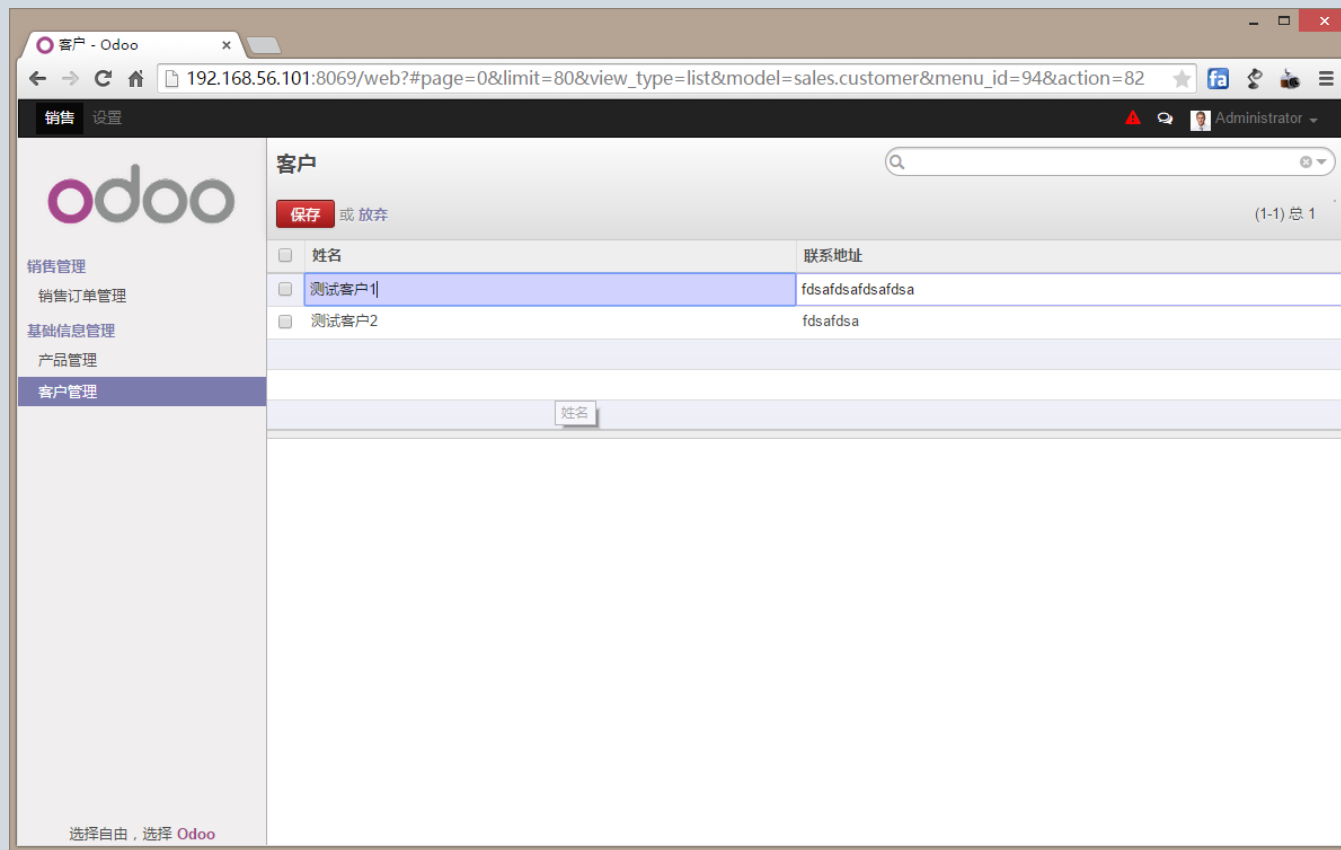
# 我们的第一个 CRUD：列表



# 我们的第一个 CRUD：表单



# 也可以直接在列表里编辑



# 实现在列表里直接编辑

---

不用定义表单视图

tree 定义里增加 editable="bottom" 即可：

- “bottom” 表示新纪录在下方新增
- “top” 表示新纪录在上方新增

```
<record id="view_sales_customer_tree" model="ir.ui.view">
  <field name="name">sales.customer.tree</field>
  <field name="model">sales.customer</field>
  <field name="arch" type="xml">
    <tree string="客户" editable="bottom">
      <field name="name" />
      <field name="address"/>
    </tree>
  </field>
</record>
```

# 我们得到了什么结果？

---

Odoo 在数据库里帮我们建好了产品表，以后修改模型 Odoo 也会帮我们自动修改表

- 增加列
- 将列设为非空
- 添加/删除索引
- etc...

没写一句 HTML/CSS/JS 就免费得到 CRUD 界面，甚至支持数据导入和记录复制

# 稍微复杂的模型： 订单

---

```
28 class Order(models.Model):
29     u'''销售订单'''
30     _name = 'sales.order'
31
32     name = fields.Char(u'单号', required=True)
33     customer = fields.Many2one('sales.customer', u'客户', required=True)
34     order_time = fields.Datetime(u'下单时间', required=True, default=fields.Datetime.now())
35     lines = fields.One2many('sales.order.line', 'order', u'订单明细')
36     price_total = fields.Float(u'总价', compute='_sum_price')
37     note = fields.Text(u'备注')
38
39     @api.one
40     @api.depends('lines')
41     def _sum_price(self):
42         price_total = 0.0
43         for d in self.lines:
44             price_total += d.subtotal
45         self.price_total = price_total
```



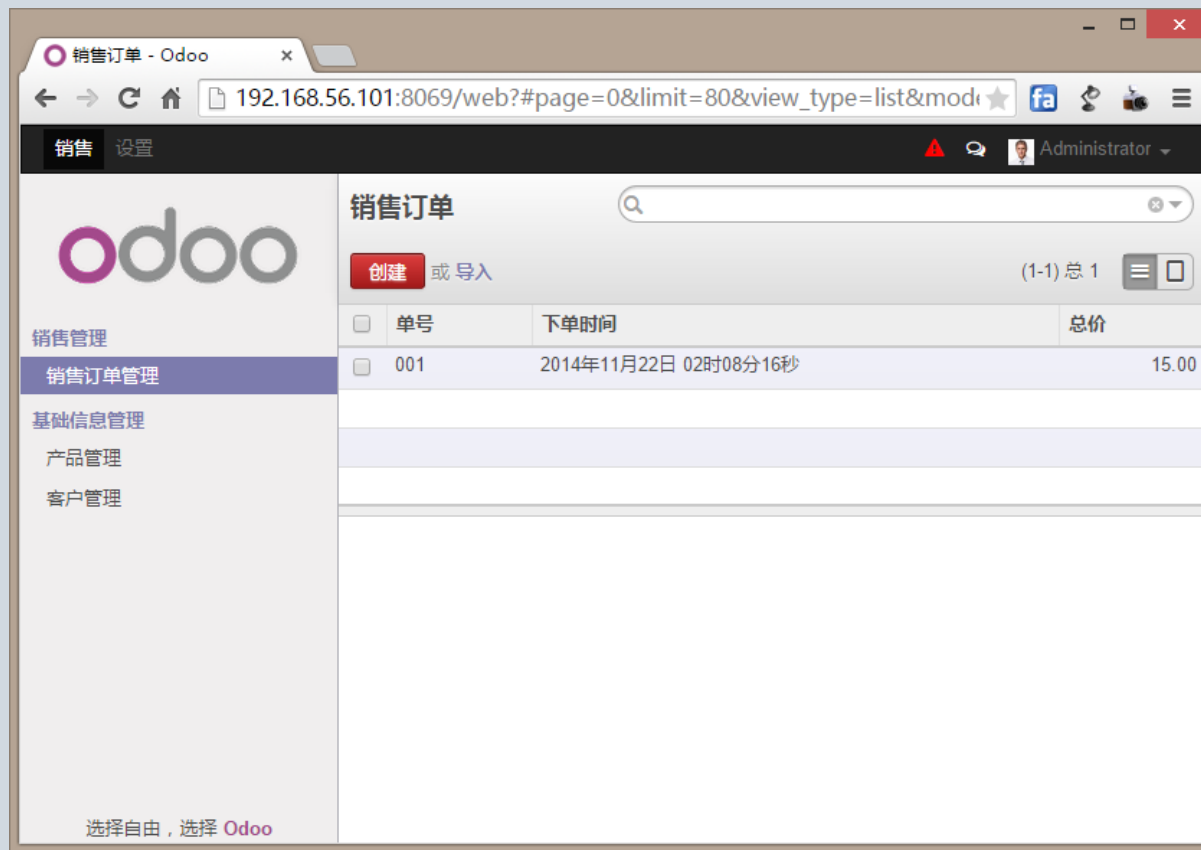
# 稍微复杂的模型： 订单明细

```
48 class OrderLine(models.Model):
49     u'''销售订单明细'''
50     _name = 'sales.order.line'
51
52     order = fields.Many2one('sales.order', u'订单', index=True, required=True, ondelete='cascade')
53     name = fields.Many2one('sales.product', u'产品', required=True)
54     quantity = fields.Float(u'数量', required=True)
55     unit_price = fields.Float(u'单价', required=True)
56     subtotal = fields.Float(u'小计', compute='_sum_subtotal')
57
58     @api.one
59     @api.depends('unit_price', 'quantity')
60     def _sum_subtotal(self):
61         self.subtotal = self.unit_price * self.quantity
62
63     @api.onchange('name')
64     def _onchange_product(self):
65         self.unit_price = self.name.unit_price
66         self.subtotal = self.unit_price * self.quantity
67
68     @api.onchange('quantity', 'unit_price')
69     def _onchange_qty_or_unit_price(self):
70         self.subtotal = self.unit_price * self.quantity
```

# 订单的表单视图定义

```
<record id="view_sales_order_form" model="ir.ui.view">
  <field name="name">sales.order.form</field>
  <field name="model">sales.order</field>
  <field name="arch" type="xml">
    <form string="销售订单">
      <sheet>
        <group>
          <field name="name" />
          <field name="customer" />
          <field name="order_time"/>
        </group>
        <group>
          <field name="price_total"/>
        </group>
        <field name="lines" nolabel="1">
          <tree editable="bottom">
            <field name="name" />
            <field name="unit_price" />
            <field name="quantity" />
            <field name="subtotal" />
          </tree>
        </field>
        <field name="note" nolabel="1" placeholder="这里添加备注..." />
      </sheet>
    </form>
  </field>
</record>
```

# 订单列表



The screenshot shows the Odoo Sales Order List interface. The browser window title is "销售订单 - Odoo". The address bar shows the URL "192.168.56.101:8069/web?#page=0&limit=80&view\_type=list&mod". The top navigation bar includes "销售" (Sales) and "设置" (Settings) tabs, and a user profile "Administrator". The left sidebar contains the Odoo logo and a menu with "销售管理" (Sales Management), "销售订单管理" (Sales Order Management), "基础信息管理" (Basic Information Management), "产品管理" (Product Management), and "客户管理" (Customer Management). The main content area is titled "销售订单" (Sales Order) and includes a search bar, a "创建" (Create) button, and a link "或导入" (or Import). Below this is a table with columns "单号" (Order Number), "下单时间" (Order Time), and "总价" (Total Price). The table contains one row with the order number "001", the order time "2014年11月22日 02时08分16秒", and the total price "15.00". The bottom of the page features the text "选择自由，选择 Odoo".

单号	下单时间	总价
001	2014年11月22日 02时08分16秒	15.00

# 订单表单

销售订单 - Odoo

192.168.56.101:8069/web?#id=2&view\_type=form&model=sales.order&action=83

销售 设置

Administrator

odoo

销售管理

销售订单管理

基础信息管理

产品管理

客户管理

选择自由, 选择 Odoo

销售订单 / 001

保存 或 放弃

单号

001

客户

测试客户1

下单时间

2014/11/22 02:08:16

总价

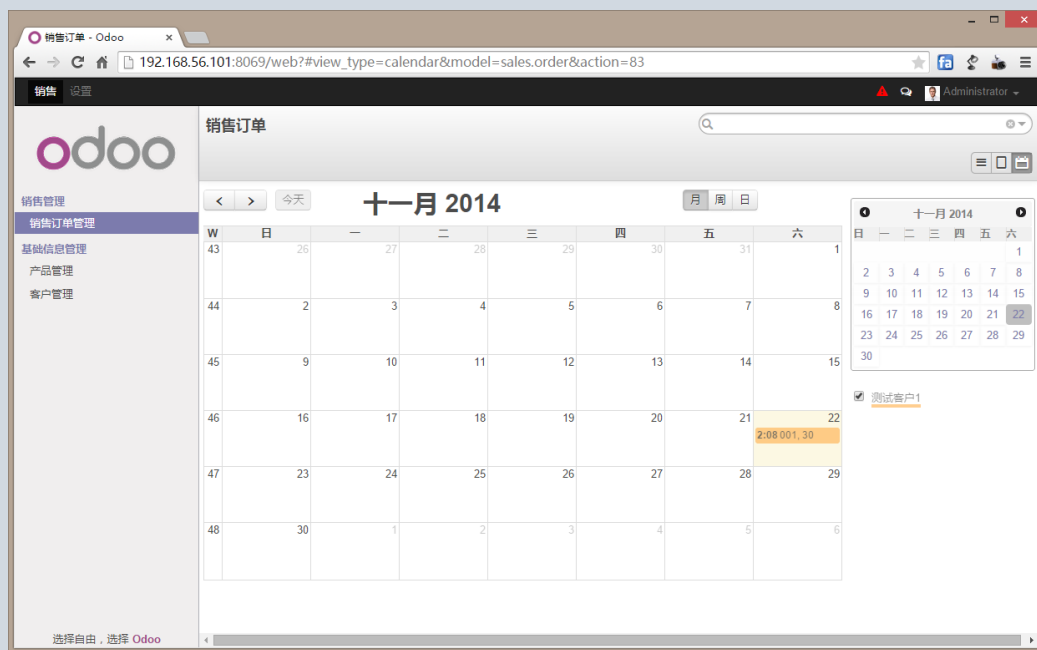
30.00

产品	单价	数量	小计	
柠檬	15.00	2.00	30.00	

添加一个项目

这里添加备注...

# 10 行代码就能增加日历视图



```
<record id="view_sales_order_calendar" model="ir.ui.view">
  <field name="name">sales.order.calendar</field>
  <field name="model">sales.order</field>
  <field name="arch" type="xml">
    <calendar string="销售订单" color="customer"
      date_start="order_time" quick_add="True" >
      <field name="name"/>
      <field name="price_total"/>
    </calendar>
  </field>
</record>
```

# 自定义界面也没问题



人力资源

IN HUMAN RESOURCE

人力资源

员工

合同

部门

员工异动

离职记录

考勤

排班表

自动排班规则

班组

班次

排班规则

排班模板

考勤

考勤机记录

排班规则 / 呼吸病区2

保存 或 放弃

2 / 4

代码

002

单位默认规则

☐

名称

呼吸病区2

单位

循环模式

按周

部门

呼吸科护理病区

忽略法定假日

☒

非工作日计入加班

☐

有效

☒

规则明细

备注

星期一	星期二	星期三	星期四	星期五	星期六	星期日
[002]付班	[002]付班	[003]夜班	[003]夜班	[001]正常班	[001]正常班	[004]休息

# ODOO 开发的优势

---

大大减小用户界面开发工作量

敏捷开发，CRUD 功能的开发时间数量级从天减为小时

数据、代码均为文本可进行版本管理，从此摆脱多人开发数据库拷来拷去的问题

全栈开发框架：

- 框架内置完善的权限系统：用户、角色（用户组）、访问控制表、菜单视图权限、记录过滤
- Code-First 的 ActiveRecord Pattern ORM
- 前台 JS 模板系统 QWeb
- 定时任务
- 报表系统
- 多语言、多货币、多时区支持
- 有限状态机工作流

# 感谢！

---

源代码仓库：<https://github.com/oldrev/odoo-dev-demo-2014>