
Sviluppo moduli OpenERP v7.0

by Dott.ssa Eliumara López

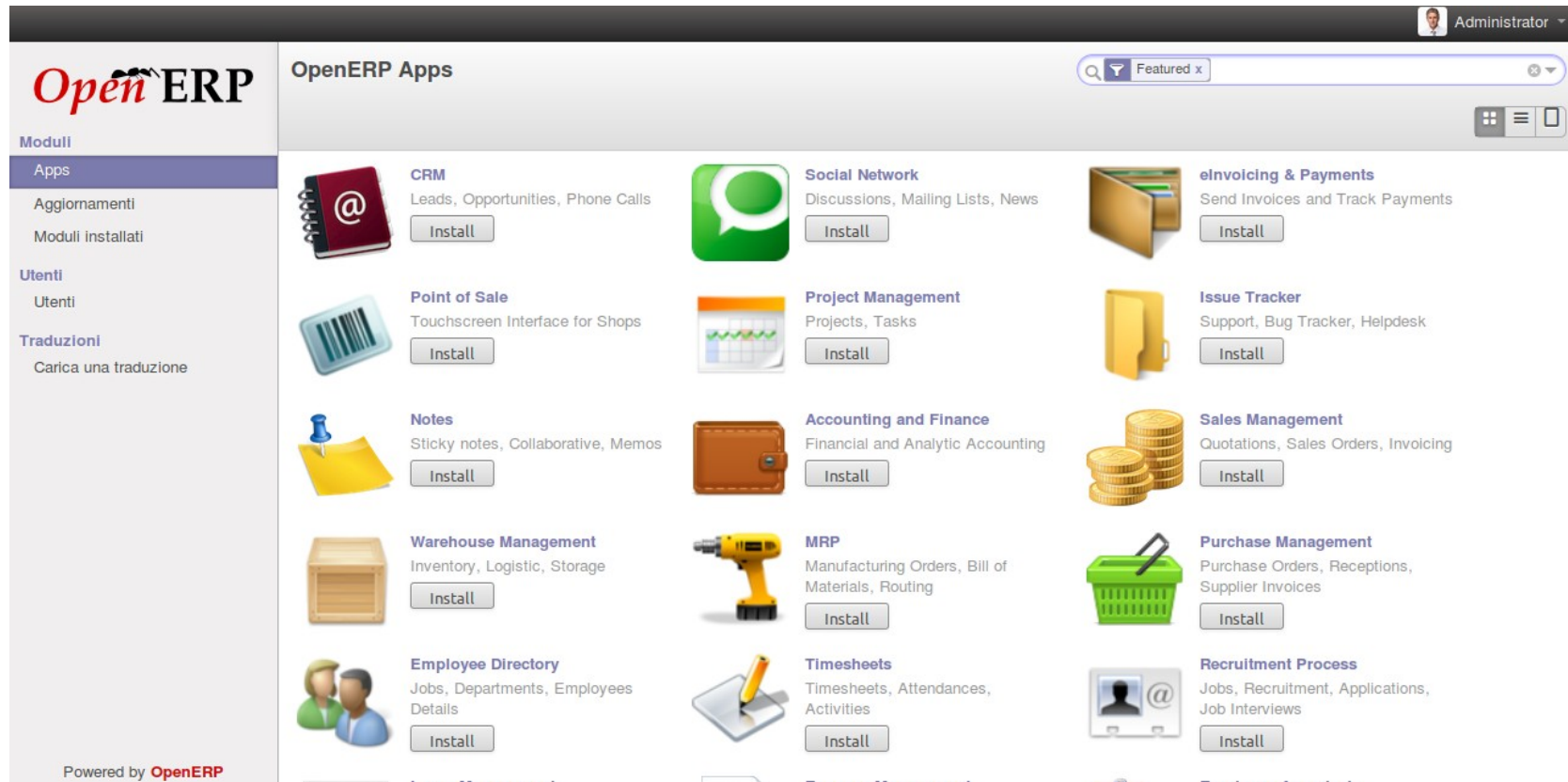
24 Ottobre 2013



- .- Cos' è un modulo?**
- .- Files**
- .- Viste, Azioni, Menu e altre interfacce**
- .- Ereditarietà**
- .- Introduzione al modulo 1**
 - * `__init__.py`
 - * `__openerp__.py`
 - * `modulo_1.py`
 - * `modulo_1_view.xml`
- .- Introduzione al modulo 2**
 - * `__init__.py`
 - * `__openerp__.py`
 - * `modulo_2.py`
 - * `modulo_2_view.xml`

Cos'è un modulo?

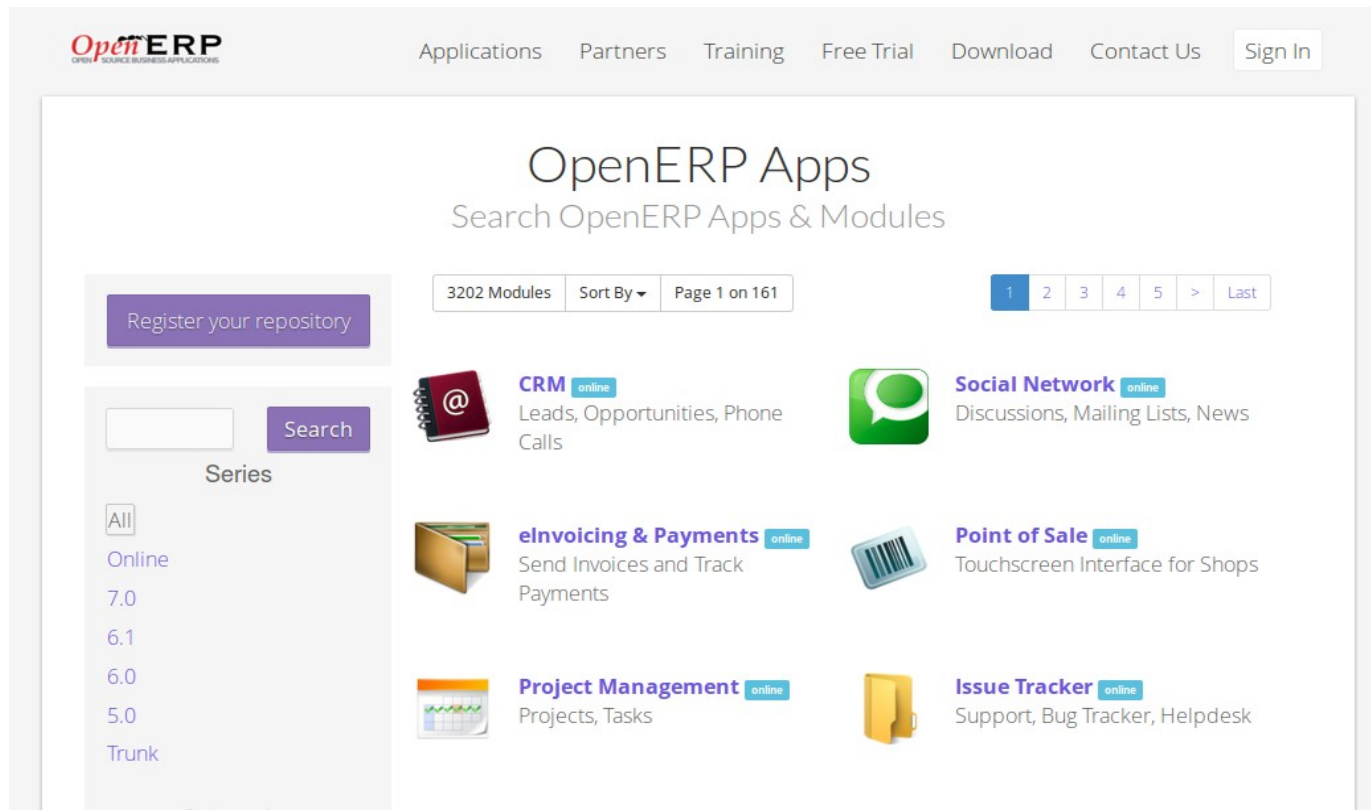
La struttura di **OpenErp** è modulare.



Un **modulo** in OpenERP è una directory che contiene dei file prestabiliti oltre a altri che variano a seconda del tipo di funzionalità implementata.

Cos'è un modulo?

Più di 1300 moduli **OpenERP** sono disponibili su <http://apps.openerp.com>



The screenshot shows the OpenERP Apps website interface. At the top, there is a navigation bar with links for Applications, Partners, Training, Free Trial, Download, Contact Us, and Sign In. The main heading is "OpenERP Apps" with the subtitle "Search OpenERP Apps & Modules". Below the heading, there is a search bar and a "Register your repository" button. The page displays a list of modules with their respective icons and descriptions:

- CRM** (online): Leads, Opportunities, Phone Calls
- Social Network** (online): Discussions, Mailing Lists, News
- Invoicing & Payments** (online): Send Invoices and Track Payments
- Point of Sale** (online): Touchscreen Interface for Shops
- Project Management** (online): Projects, Tasks
- Issue Tracker** (online): Support, Bug Tracker, Helpdesk

The page also shows a pagination bar indicating "3202 Modules" and "Page 1 on 161".

I files all'interno di un **modulo** in OpenERP essenzialmente sono:



mio_modulo_view.
xml



__init__.py



mio_modulo.py



__openerp__.py

Alcuni moduli sono più robusti e hanno altre sottocartelle come:
report, edi, wizard, process, ecc...

File `__init__.py`

Contiene un comando di import del modulo stesso.
Permette caricare il modulo creato.

File `__openerp__.py`

Contiene un dizionario python, dove vengono descritte le funzionalità, dipendenze implementate dal modulo e molto altro ancora.

File `__openerp__.py`

Le chiavi del dizionario sono:

```
'name'  
'description'  
'version'  
'author'  
'website'  
'category'  
'depends'  
'installable'  
'active'  
'init_xml'  
'update_xml'  
'demo_xml'  
...
```


File mio_modulo.py

In questo file sono definiti gli oggetti che compongono le view del modulo e quelli del database.

File mio_modulo.py

Attributi degli oggetti:

```
_name  
_description (facoltativo)  
_inherit (facoltativo)  
_columns  
_constraints (facoltativo)  
_sql_constraints (facoltativo)  
_defaults  
_order (facoltativo)  
_log_access
```

File mio_modulo.py

Tipi dati:

boolean: assume valore True o False

Esempio:

```
'active': fields.boolean('Active')
```

integer integer_big: valori interi positivi o negativi

Esempio:

```
'id': fields.integer('Inventory Line Id', readonly=True)
```

selection: lista di valori che il campo può assumere

Esempio: `fields.selection`

```
[('draft', 'Draft'),  
 ('open', 'Opened'),  
 ('close', 'Accepted'), ('cancel', 'Canceled')],  
 'Status', readonly=True)
```

float: valori decimali

char: stringa con una dimensione massima

text: stringa senza dimensione massima

Esempio:

```
'note': fields.text('Description', translate=True)
```

date: data

...

File mio_modulo.py

Tipi dati:

many2one

Esempio:

```
'category_id': fields.many2one('idea.category', 'Category',  
    required=True )
```

one2many

Esempio:

```
'vote_ids' : fields.one2many('idea.vote', 'idea_id', 'Vote')
```

many2many

Esempio:

```
'category_id': fields.many2many('res.partner.category',  
    'res_partner_category_rel', 'partner_id',  
    'category_id', 'Categories')
```

File mio_modulo_view.xml

Interfacce utente per gestire gli oggetti definiti nel model.

```
<?xml version="1.0" encoding="utf-8"? >  
<openerp >  
  <data >  
  
    [view definitions]  
  
  </data >  
</openerp >
```

Viste, Azioni, Menu e altre interfacce

Tipologie di Viste:

- ✓ form view
- ✓ tree view
- ✓ search view
- ✓ graph (grafici)
- ✓ gantt (Diagramma di gantt)
- ✓ calendar (calendario)
- ✓ ...

Viste, Azioni, Menu e altre interfacce

Struttura XML delle viste

```
<record model="ir.ui.view"
  id="identificativo_univoco_della_vista">
  <field name="name">nome.vista</field>
  <field name="model">nome_oggetto</field> # oggetto sul
  quale la vista è definita
  <field name="type">form</field> # tree, form,
  calendar, search, graph, gantt
  <field name="arch" type="xml"> # architettura della
  vista
    <!-- architettura e definizione della vista:
  <form>, <tree>, <graph>, ... -->
  </field>
</record>
```

Viste, Azioni, Menu e altre interfacce

Azioni

Azioni che determina il comportamento del sistema a seguito di un evento.

Struttura XML delle azioni

```
<!-- Action -->
  <record model="ir.actions.act_window" id="nome_uniovoco">
    <field name="name">nome_desc</field>
    <field name="res_model">nome.modello</field>
    <field name="view_type">tipo_vista</field>
  </record>
```


Viste, Azioni, Menu e altre interfacce

Menu

Azioni che determina il comportamento del sistema a seguito di un evento.

Struttura XML dei menu

```
<menuitem name="nome_menu"  
  parent="base.menu_tools" id="nome_menu1" sequence="4"/>
```

```
<menuitem  
  name="nome_menu" parent="nome_menu1"  
  id="nome_menu_tree"  
  action="action_nome_menu_tree"/>
```

Inherit Model

```
_inherit='object.name'
```

Inherit for Extension (`_name == _inherit`):

```
class res_partner(osv.osv):  
    _name = 'res.partner'  
    _inherit="res.partner"  
    _columns = {  
        'codcompanyclient': fields.integer('Code Company  
        Client', size=4),  
        'nit': fields.char('NIT', size=10),  
        'disp': fields.boolean('Disponibilita'),  
        'es_impiegato': fields.boolean('Es Impiegato'),  
    }  
    res_partner()
```

Inheritance by prototyping (_name != _inherit):

```
class other_material(osv.osv):
    _name = 'other.material'
    _inherit = 'network.material'
    _columns = {
        'manuf_warranty': fields.boolean('Manufacturer
            warranty?'),
    }
    _defaults = {
        'manuf_warranty': lambda *a: False,
    }
    other_material()
```

Inheritance by Delegation:

```
class tiny_object(osv.osv)
_name = 'tiny.object'
_table = 'tiny_object'
_inherits = {
'tiny.object_a': 'object_a_id',
'tiny.object_b': 'object_b_id',
... ,
'tiny.object_n': 'object_n_id'
}
(...)
```

Inherit View

```
<record model="ir.ui.view" id="view_partner_form">
  <field name="name">res.partner.form.inherit</field>
  <field name="model">res.partner</field>
  <field name="inherit_id" ref="base.view_partner_form"/>
  <field name="type">form</field>
  <field name="arch" type="xml">
    <notebook position="inside">
      <page string="Relations">
        <field name="relation_ids" colspan="4" nolabel="1"/>
      </page>
    </notebook>
  </field>
</record>
```

Siti consigliati e Contatti

openerp-italia.org



openerp.com



[eliuimara.lopez \(skype\)](https://www.skype.com/people/eliuimara.lopez)

[@elilopezlopez \(twitter\)](https://twitter.com/elilopezlopez)

lopez@cecchi.info