

# Développement d'un module

Structure d'un module

Le modèle de données

Définition des vues et des menus

# Structure d'un module

- Un module est un package python qui peut contenir:
  - Objets métier : classes python
  - Fichiers de données (xml ou json): métadonnées pour les vues, workflows et fichiers de configuration
  - Contrôleurs web:
  - Fichiers statiques: images, css, javascripts
- Un module est un dossier stocké dans un dossier de modules (spécifié par l'option `--addons-path` )
  - Fichier obligatoires
    - Manifest: `__openerp__.py`
    - `__init__.py`

# Création d'un module

- Ouvrez une invite de commande en mode administrateur
- Lancez la commande suivante à partir du dossier installation\_odoo\server

```
C:\Odoo\server>odoo scaffold gprojets openerp/addons
```

- Odoo crée la structure de base d'un module

OS (C:) > Odoo > server > openerp > addons > gprojets

Nom	Modifié le	Type	Taille
security	18/01/2015 15:17	Dossier de fichiers	
__init__	18/01/2015 15:17	Fichier PY	1 Ko
__openerp__	18/01/2015 15:17	Fichier PY	1 Ko
controllers	18/01/2015 15:17	Fichier PY	1 Ko
demo	18/01/2015 15:17	Fichier XML	1 Ko
models	18/01/2015 15:17	Fichier PY	1 Ko
templates	18/01/2015 15:17	Fichier XML	1 Ko

## Le fichier manifest ( \_\_ openerp \_\_ .py)

La présence du fichier \_\_openerp\_\_.py est obligatoire dans un module, il contient des métadonnées sur le module (nom, version, description, auteur ...) et la liste des fichiers nécessaire pour le bon fonctionnement du module.

```
{
    'name': "gprojets",
    'category': 'Uncategorized',
    'version': '0.1',
    # Liste des modules obligatoires
    'depends': ['base'],

    # toujours chargés
    'data': ['templates.xml',
    ],
    # chargés uniquement en mode démonstration
    'demo': [
        'demo.xml',
    ],
}
```

## Le fichier \_\_init\_\_.py

Un module odoo est aussi un package python, le fichier \_\_init\_\_.py doit contenir les instructions import .

```
- .....
import controllers
import models
```

# Le modèle

- Les objets métiers Odoo sont des classes qui héritent de la classe Model définie dans le package models, toutes les classes de type Model doivent au moins avoir l'attribut `_name` qui définit le nom du modèle dans Odoo.
  - Exemple:

## Référence

<https://www.odoo.com/documentation/8.0/howtos/backend.html#build-an-odoo-module7>

### Fichier models.py

```
from openerp import models, fields, api
class Projet (models.Model):
    _name='gprojets.projet'
    #la propriété string définit l'étiquette du
    champ, par défaut elle est identique au nom du
    champ.

    name=fields.Char(string='Nom',required=True)
    # la propriété description s'affichera dans
    une zone de texte multilingue
    description=fields.Text()
```

# Les actions et les menus (gprojets.xml)

- Créer la vue gprojets/views/gprojets.xml
- Ajouter une référence vers ce fichier dans le le manifest, section data: 'views/gprojets.xml',

```
<?xml version="1.0" encoding="utf-8"?>
<openerp>
  <data>
    <!-- window action -->
      <menuitem name="Menu Projets" id="menu_root" />
    <!--Menu gauche-->
    <!--
      le record suivant est la définition d'une
      action de type window,
      c'est une action qui ouvre une vue ou un
      ensemble de vues
      -->
    <record model="ir.actions.act_window"
id="action_projets">
      <field name="name">Projets</field>
      <field name="res_model">gprojets.projet</field>
      <field name="view_mode">tree,form</field>
    </record>

    <menuitem name="General" id="menu_general"
parent="menu_root"/>

    <menuitem name="Projets" id="menu_projets"
parent="menu_general"
      sequence="1" action="action_projets"/>
  </data>
</openerp>
```

```
<!-- Ajout d'un formulaire-->
<record model="ir.ui.view" id="projet_form">
  <field name="name">vue projet</field>
  <field name="model">gprojets.projet</field>
  <field name="arch" type="xml">

    <form string="Projet">
      <sheet>
        <h1>
          <field name="name" placeholder="Nom du projet"/>
        </h1>
        <notebook>
          <page string="Description">
            <field name="description"/>
          </page>
        </notebook>
      </sheet>
    </form>

  </field>
</record>
</data>
</openerp>
```

# Données de démonstration: demo.xml

```
<openerp>
  <data>
    <record model="gprojets.projet"
id="projet0">
      <field name="name">projet 0</field>
      <field name="description">description
projet

peut avoir plusieurs lignes
      </field>
    </record>
    <record model="gprojets.projet"
id="projet1">
      <field name="name">projet 1</field>
      <!-- no description for this one -->
    </record>
```

```
      <record model="gprojets.projet" id="projet2">
        <field name="name">projet 2</field>
        <field
name="description">description projet 2</field>
      </record>
    </data>

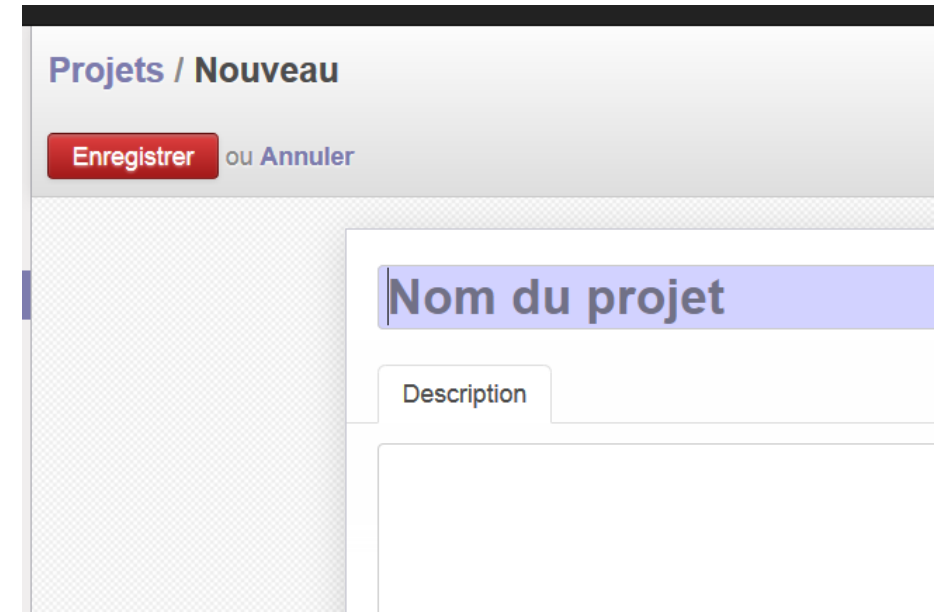
</openerp>
```

# Le module après installation

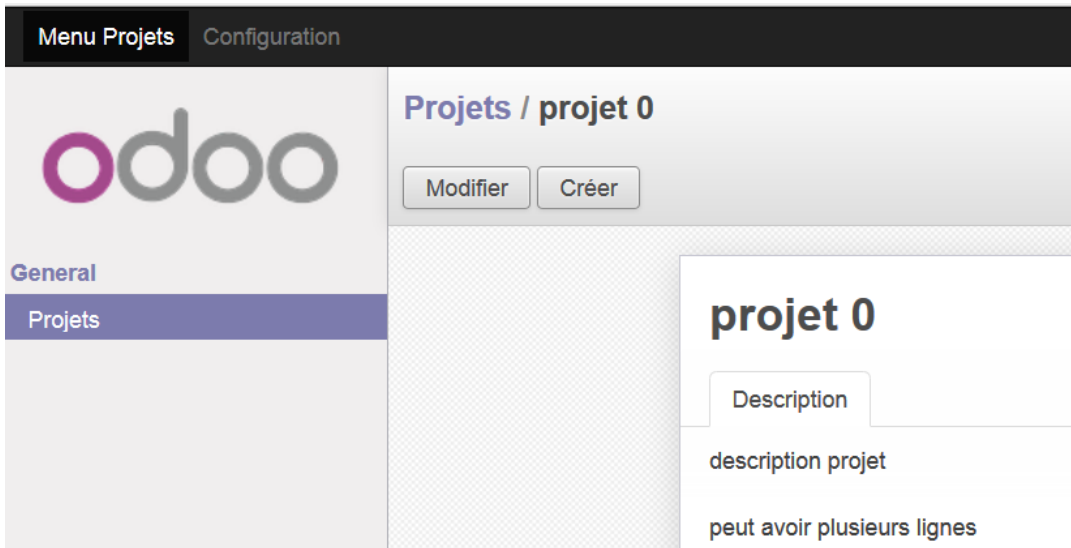
## Le module Projets



## Ajout d'un projet



## Détail d'un projet



# Création de la vue Projet.

- La vue Projet doit afficher

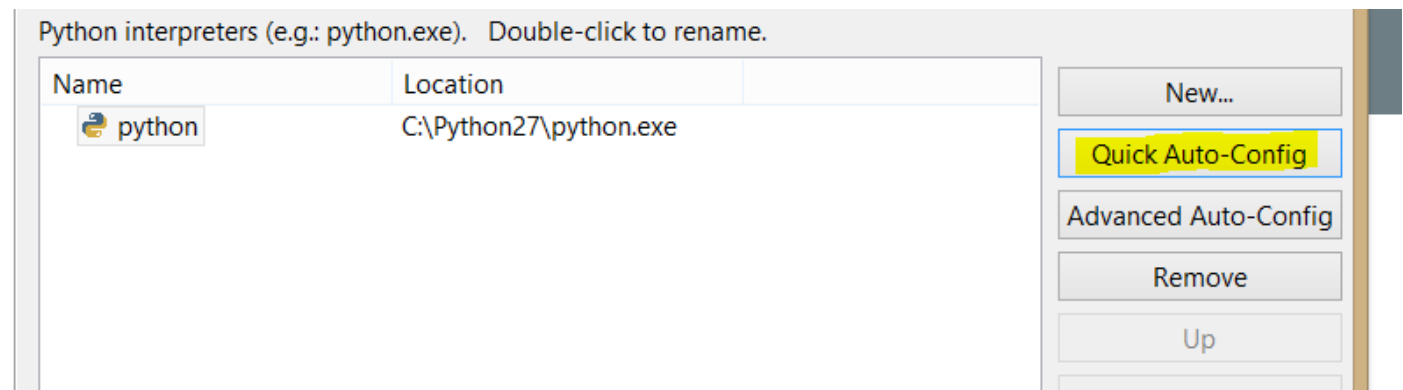
# Configuration PyDev Eclipse

- Python
  - <https://www.python.org/ftp/python/2.7.9/python-2.7.9.msi>
- PyDev
  - <http://pydev.org/updates>
- Egit
  - <http://download.eclipse.org/egit/updates>
    - Cocher: Eclipse Git Team Provider
- Télécharger les templates pour odoo
  - <http://openerp-eclipse-template.googlecode.com/svn/trunk/>
    - openerp-eclipse-xml-template.xml
    - templates-openerp.xml

# Configuration python

- d

Interactive Console  
Interpreters  
IronPython Interpreter  
Jython Interpreter  
Python Interpreter  
Logging



- > Mylyn
- > Plug-in Development
- ▲ PyDev
  - Builders
  - ▲ Editor
    - Auto Imports
    - Code Analysis
    - Code Completion
    - Code Completion (ctx insensitive and common tokens)
    - Code Folding
    - > Code Style
    - Editor caption/icon
    - Hover
    - Mark Occurrences
    - Overview Ruler Minimap
    - Save Actions
    - Templates
    - Typing
    - Vertical Indent Guide
  - > Interactive Console
  - > Interpreters
  - Logging

Name	Context	Description
<input checked="" type="checkbox"/> <u>_terp_</u>	<u>Editor</u>	<u>_terp_</u>
<input checked="" type="checkbox"/> <u>_columns</u>	Editor	<u>_columns</u>
<input checked="" type="checkbox"/> <u>_constraints</u>	Editor	<u>_constraints</u>
<input checked="" type="checkbox"/> <u>_date_name</u>	Editor	<u>_date_name</u>
<input checked="" type="checkbox"/> <u>_defaults</u>	Editor	<u>_defaults</u>
<input checked="" type="checkbox"/> <u>_description</u>	Editor	<u>_description</u>
<input checked="" type="checkbox"/> <u>_inherit</u>	Editor	<u>_inherit</u>
<input checked="" type="checkbox"/> <u>_inherits</u>	Editor	<u>_inherits</u>
<input checked="" type="checkbox"/> <u>_name</u>	Editor	<u>_name</u>
<input checked="" type="checkbox"/> <u>_order</u>	Editor	<u>_order</u>
<input checked="" type="checkbox"/> <u>_parent_store</u>	Editor	<u>_parent_store</u>
<input checked="" type="checkbox"/> <u>_rec_name</u>	Editor	<u>_rec_name</u>
<input checked="" type="checkbox"/> <u>_sequence</u>	Editor	<u>_sequence</u>
<input checked="" type="checkbox"/> <u>_sql_constraints</u>	Editor	<u>_sql_constraint</u>
<input checked="" type="checkbox"/> <u>_table</u>	Editor	<u>_table</u>
<input checked="" type="checkbox"/> <u>&lt;Fmnty&gt;</u>	New Module	Module: Fmnty

Buttons: New..., Edit..., Remove, Restore Removed, Revert to Default, **Import...**, Export...

Preview:

```
"version": "1.0",
"depends": ["base"]
```

- templates-openerp

- XML
  - > DTD Files
  - XML Catalog
  - XML Files
    - Editor
      - Content Assist
      - Syntax Coloring
      - Templates
      - Typing
      - Validation

- Openerp-eclipse-xml-template.xml

Configuration python

- Télécharger justify-nav.css dans le dossier Content
  - <http://getbootstrap.com/examples/justified-nav/justified-nav.css>
- Ajoutez justify-nav.css dans le bundle ~Content/css
- Supprimer le div qui a la classe class="navbar navbar-inverse navbar-fixed-top"