onnet

2021
VIE

Customizing Odoo Base Addons

www.on.net.vn
www.arrowhitech.com

Table of contents

onnet
BY AHT TECH JSC

About Customization

odoo.conf

addon_paths=

- odoo/addons
- enterprise/
- **project/addons/custom**

You should never modify odoo base code

Instead, you create new custom addon, make it depend on base addon(s)

__manifest__.py

'depends': ['base']

Customizing = Inheriting + Adding new things/logic

About Customization

Module dependency

A dependency means that the Odoo framework will ensure that these modules are installed before our module is installed.

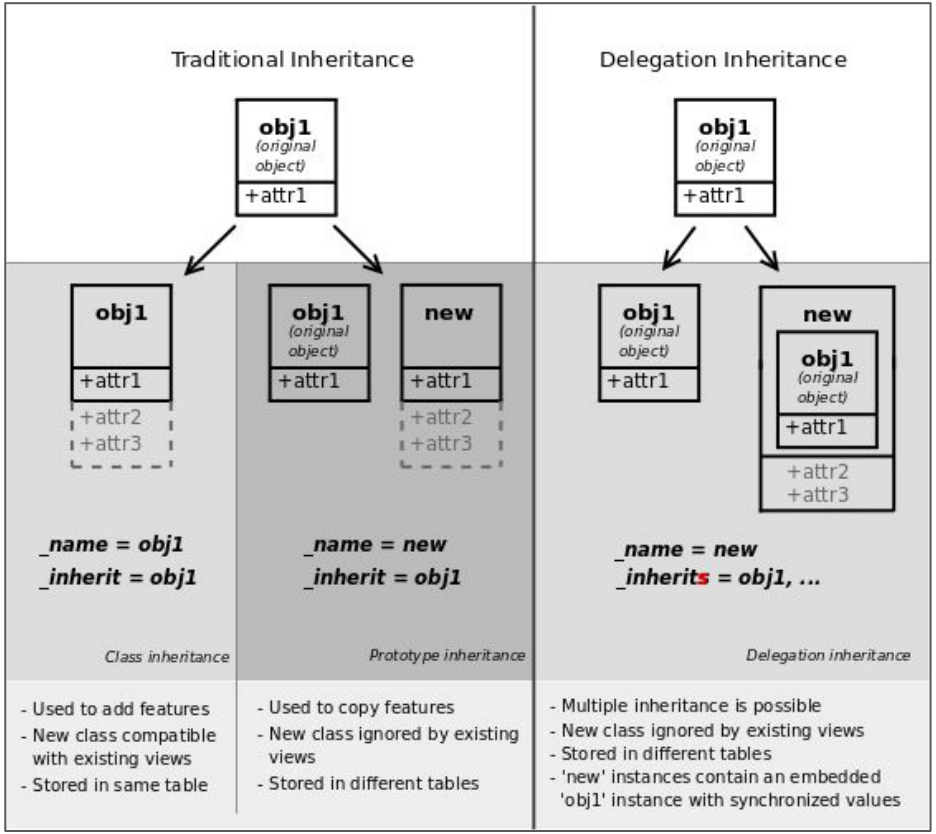
Moreover, if one of these dependencies is uninstalled, then our module and any other that depends on it will also be uninstalled.

Module dependency is also ‘inheritable’

Model Inheritance

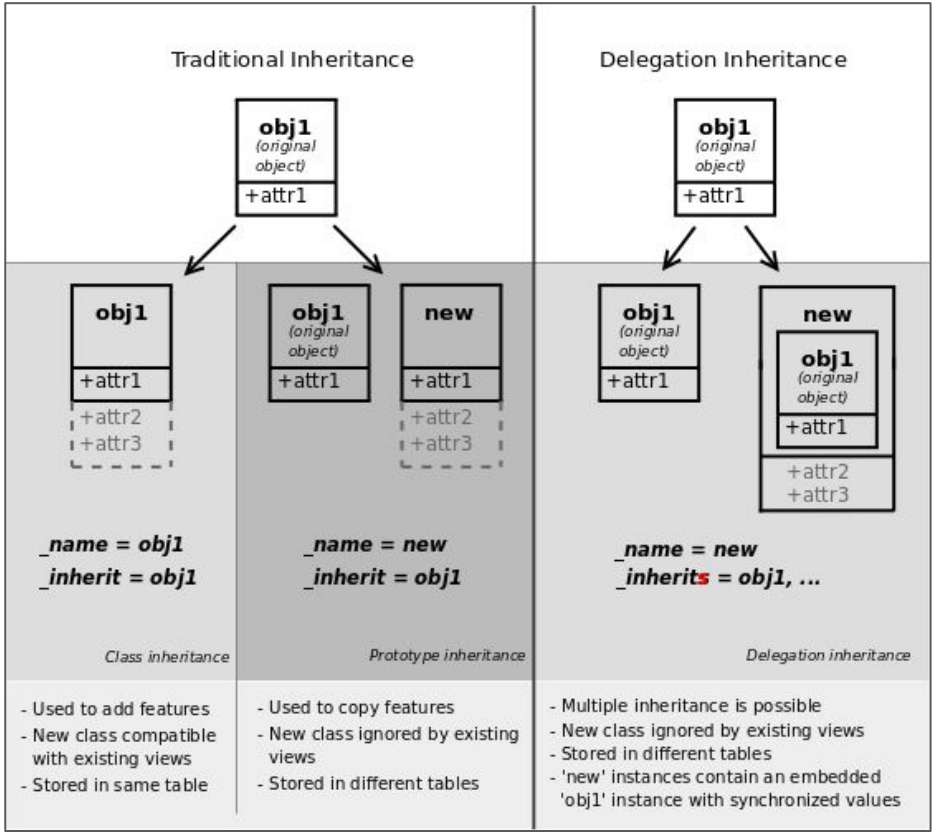
The first inheritance mechanism allows a module to modify the behavior of a model defined in another module:

- add fields to a model,
- override the definition of fields on a model,
- add constraints to a model,
- add methods to a model,
- override existing methods on a model.



Model Inheritance

The second inheritance mechanism (delegation) allows to link every record of a model to a record in a parent model, and provides transparent access to the fields of the parent record.



Model Inheritance

odoo Gold Partner

Extend business logic defined in a model

```
class Partner(models.Model):
    _inherit = 'res.partner'

    linkedin = fields.Char('LinkedIn', help="LinkedIn of this contact")
    facebook = fields.Char('Facebook', help="Facebook of this contact")

    @api.model
    def create(self, values):
        # Do something here
        return super(Partner, self).create(values)
```

onnet BY AHT TECH JSC

View Inheritance

Inheritance Specs - 3 types:

- An **xpath** element with an **expr** attribute.
- A field element with a name attribute, matches the first field with the same name.
- Any other element: the first element with the same name and identical attributes (ignoring position and version attributes) is matched

```
<xpath expr="page[@name='pg']/group[@name='gp']/field" position="inside">
  <field name="description"/>
</xpath>
<field name="res_id" position="after"/>
<div name="name" position="replace">
  <div name="name2">
    <field name="name2"/>
  </div>
</div>
```

View Inheritance

Optional position attribute specifying how the matched node should be altered

- inside (default)
- replace
- after
- before
- attributes
- move

```
<field name="sale_information" position="attributes">
  <attribute name="invisible">0</attribute>
  <attribute name="attrs">
    {'invisible': [('sale_ok', '=', False)], 'readonly': [('editable', '=', False)]}
  </attribute>
</field>
```

View Inheritance

odoo Gold Partner

Display new fields

```
<record id="view_partner_form" model="ir.ui.view">
      <field name="name">view.res.partner.form.custom</field>
      <field name="model">res.partner</field>
      <field name="inherit_id" ref="base.view_partner_form"/>
      <field name="arch" type="xml">
          <xpath expr="//field[@name='website']" position="after">
              <field name="linkedin" widget="url"/>
              <field name="facebook" widget="url"/>
          </xpath>
      </field>
  </record>
```

onnet BY AHT TECH JSC

Template Inheritance

```
<template id="product_wishlist" inherit_id="website_sale_wishlist.product_wishlist">
    <xpath expr="//button[hasclass('o_wish_rm')]" position="after">
        <button type="button" class="btn btn-link o_add_to_compare no-decoration" t-att-data-product-id='wish.product_id.id'>
            <small><i t-attf-class="fa fa-exchange"></i> Add to compare</small>
        </button>
    </xpath>
</template>
```

JS Customization

- First of all, remember that the first rule of customizing odoo with JS is: **try to do it in python**.
- What you can do:
  - Modifying an existing widget
  - Customizing an existing view
    - extending view
    - add it to view registry
    - use it in actual view

```
var HelpdeskDashboardRenderer = KanbanRenderer.extend({
    ...
});

var HelpdeskDashboardModel = KanbanModel.extend({
    ...
});

var HelpdeskDashboardController = KanbanController.extend({
    ...
});

var HelpdeskDashboardView = KanbanView.extend({
    config: _.extend({}, KanbanView.prototype.config, {
        Model: HelpdeskDashboardModel,
        Renderer: HelpdeskDashboardRenderer,
        Controller: HelpdeskDashboardController,
    }),
});
```

JS Customization

- First of all, remember that the first rule of customizing odoo with JS is: **try to do it in python**.
- What you can do:
  - Modifying an existing widget
  - Customizing an existing view
    - extending view
    - add it to view registry
    - use it in actual view

```
var viewRegistry = require('web.view_registry');
viewRegistry.add('helpdesk_dashboard', HelpdeskDashboardView);
```

JS Customization

- First of all, remember that the first rule of customizing odoo with JS is: **try to do it in python**.
- What you can do:
  - Modifying an existing widget
  - Customizing an existing view
    - extending view
    - add it to view registry
    - use it in actual view

```
<record id="helpdesk_team_view_kanban" model="ir.ui.view" >
    ...
    <field name="arch" type="xml">
        <kanban js_class="helpdesk_dashboard">
            ...
        </kanban>
    </field>
</record>
```

JS Customization

- First of all, remember that the first rule of customizing odoo with JS is: **try to do it in python**.
- What you can do:
  - Modifying an existing widget
  - Customizing an existing view
    - extending view
    - add it to view registry
    - use it in actual view
  - Extend client-side qweb template

```
<t t-name="child.template" t-inherit="base.template"
t-inherit-mode="primary">
    <xpath expr="//ul" position="inside">
        <li>new element</li>
    </xpath>
</t>

<t t-inherit="base.template" t-inherit-mode="extension">
    <xpath expr="//tr[1]" position="after">
        <tr><td>new cell</td></tr>
    </xpath>
</t>
```

JS Customization

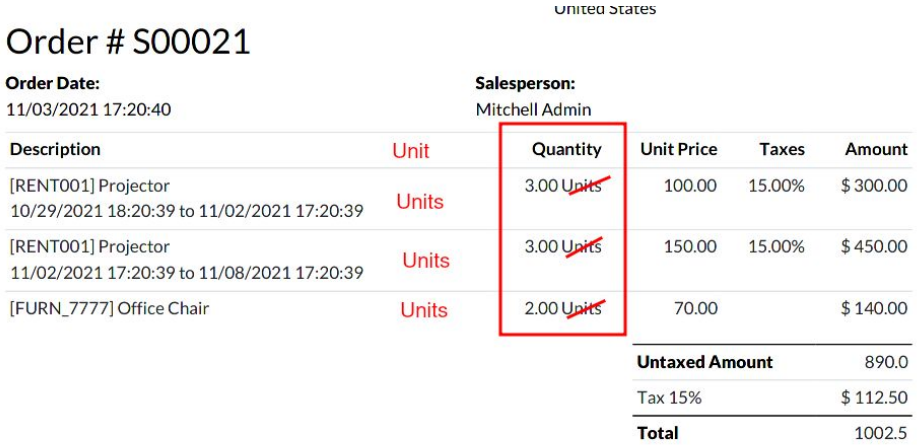
- First of all, remember that the first rule of customizing odoo with JS is: **try to do it in python**.
- What you can do:
  - Modifying an existing widget
  - Customizing an existing view
    - extending view
    - add it to view registry
    - use it in actual view
  - Extend client-side qweb template
  - Patching code: Odoo provides the utility function patch. It is mostly useful to override/update the behavior of some other component/piece of code that one does not control.

Practicals

Customize Calendar Module

- Add new field Many2one ‘Room’ for Meeting (calendar.event)
- Show the new field in calendar, form, list view

Customize Sale Report



United States

Order # S00021

Order Date:
11/03/2021 17:20:40

Salesperson:
Mitchell Admin

Description	Unit	Quantity	Unit Price	Taxes	Amount
[RENT001] Projector 10/29/2021 18:20:39 to 11/02/2021 17:20:39	Units	3.00 Units	100.00	15.00%	$ 300.00
[RENT001] Projector 11/02/2021 17:20:39 to 11/08/2021 17:20:39	Units	3.00 Units	150.00	15.00%	$ 450.00
[FURN_7777] Office Chair	Units	2.00 Units	70.00		$ 140.00

Untaxed Amount	890.0
Tax 15%	$ 112.50
Total	1002.5

Q&A