Easy PDF report creator module

This module allows us to create whatever model report you want that transmitting PDF report using few codes and much easy way. Now we'll work on ODOO v9 and v10. While working these versions we'll find out the following advantages.

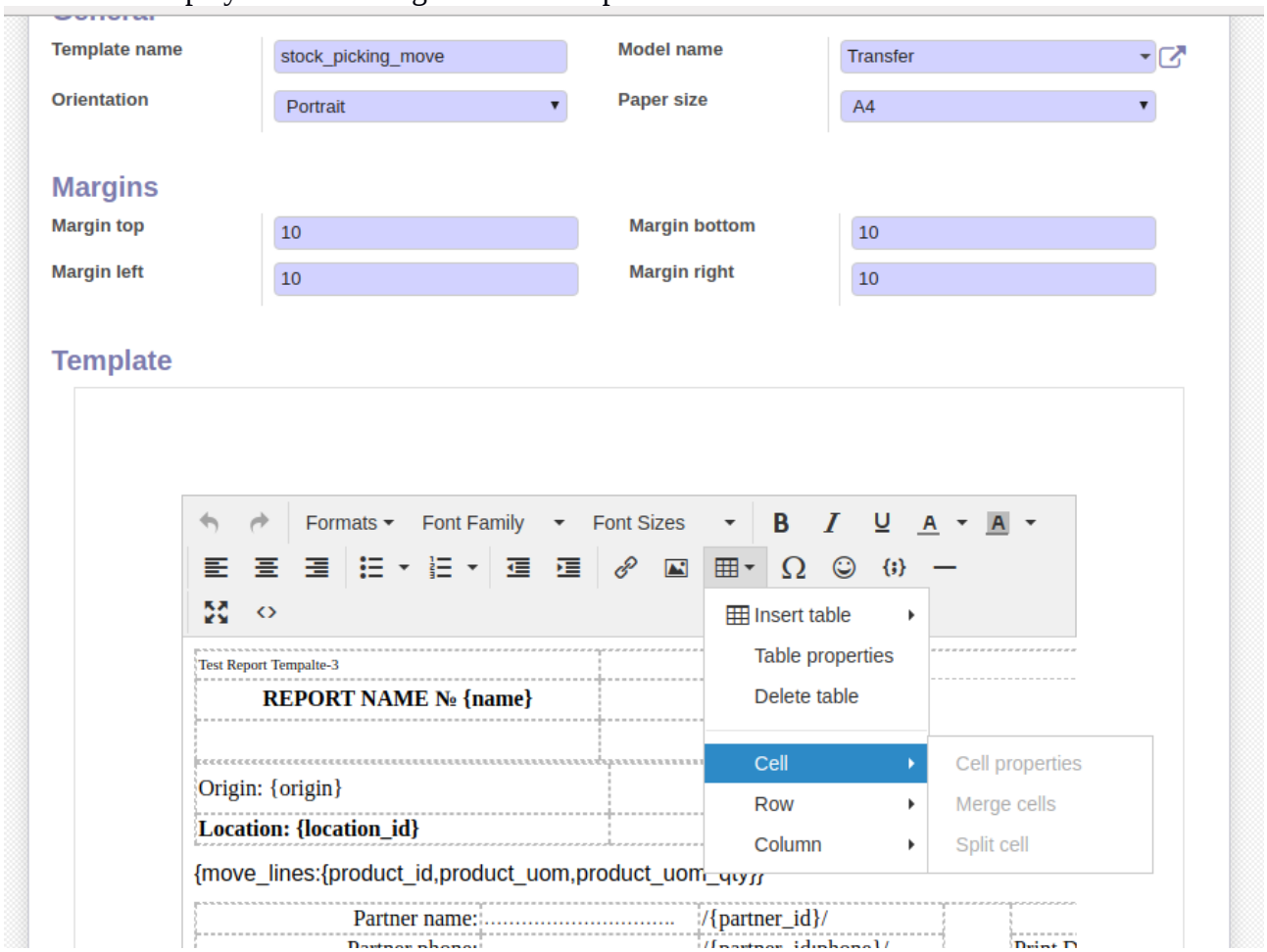
Advantages:

- No need of high knowledge of development (very few .PY codes will be written, but not to draw any model at .XML).
- Easy to prepare report model. Work on HTML rich text editor (likely MS word)
- Allows to create any model and report adjustment on any chosen model
- All models shall be added default menu 'EasyPrint' on Sidebar menu. You only need to draw your template designs.
- It shows a drawing on the model in any size and drawing from URL
- It gives a chance to merge, split, resize the Table
- Create PDF in a short time and any desired model

It is described below using a example on stock.picking model.

Instructions

After installing Module ODOO to be loaded on Developer mode then Setting→Reports→Easy PDF creator menu appears on the screen. Here we should identify the report model.
The screen displays the followings and most important 2 fields are described as below:

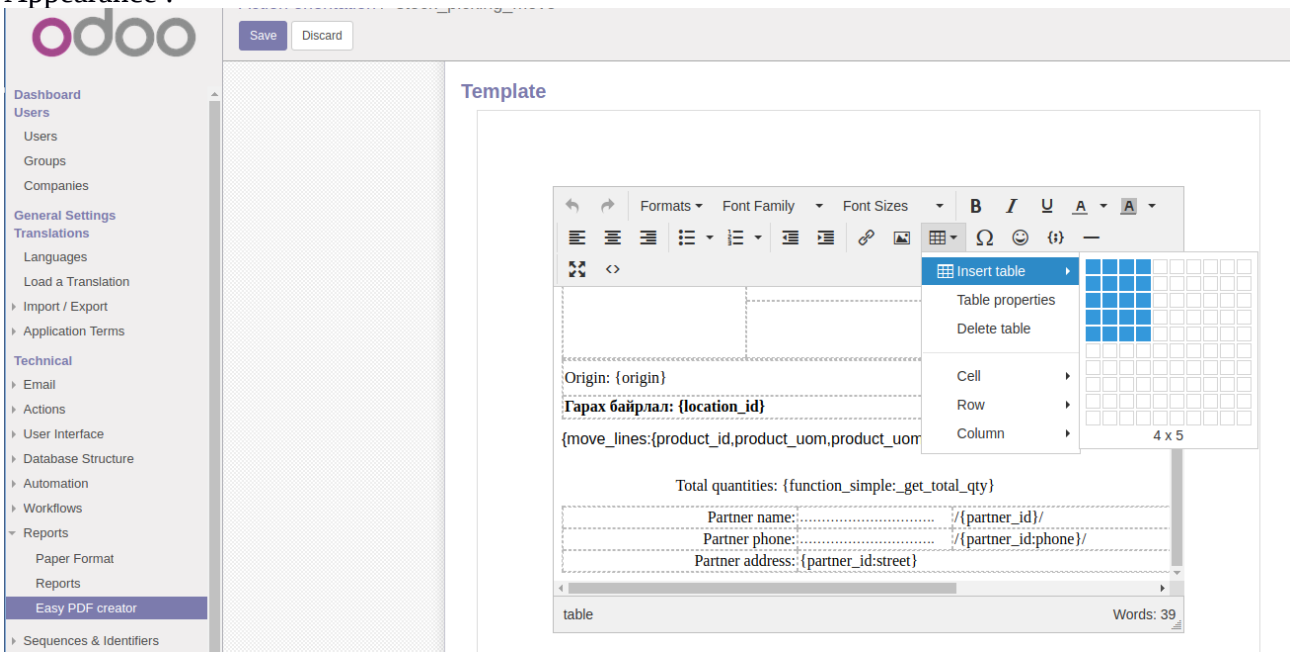


Field description:

- Template name – Must be unique. This name will be used to find an adjusted report on the module.
- Model name – It describes report model where it adjusted.

Create a model

Appearance :



It is very easy to prepare and adjust a model. Because we use HTML rich text editor. It is the same of MS word. There are two types: Constant text and variable (field's value)
- Constant text should be written directly.
- Any types of field's value located on the model will be adjusted as below (syntax)

- Integer, Char, Float, Text, Date fields: These fields name will be written {}in brackets. Syntax: *{field_1}, {field_2}* etc. EG: {name}, {origin}

- Many2one field: {} will be written in the brackets. As well as we can approach model field of Many2one field. So that : colon will be written too. Syntax: {many2one_field}, *{many2one_field:field_1}* . Example: {partner_id}, {partner_id:phone}, {partner_id:street}

- One2many field: It will be written as below. Syntax: *{one2many_field:{field1,field2,field3}}* Example: **product name, quantity, unit, price unit** of move_lines of Stock.picking model are adjusted to be printed out.
  {move_lines:{product_id,product_uom,product_uom_qty,price_unit}}

- Call function: It allows us to print out transmitted value using call function from the model. Function return value must be the types of: **text, int, float.** Syntax: *{function_simple:your_function_name}* . Example: {function_simple:_get_total_qty}
  function example:
  @api.multi

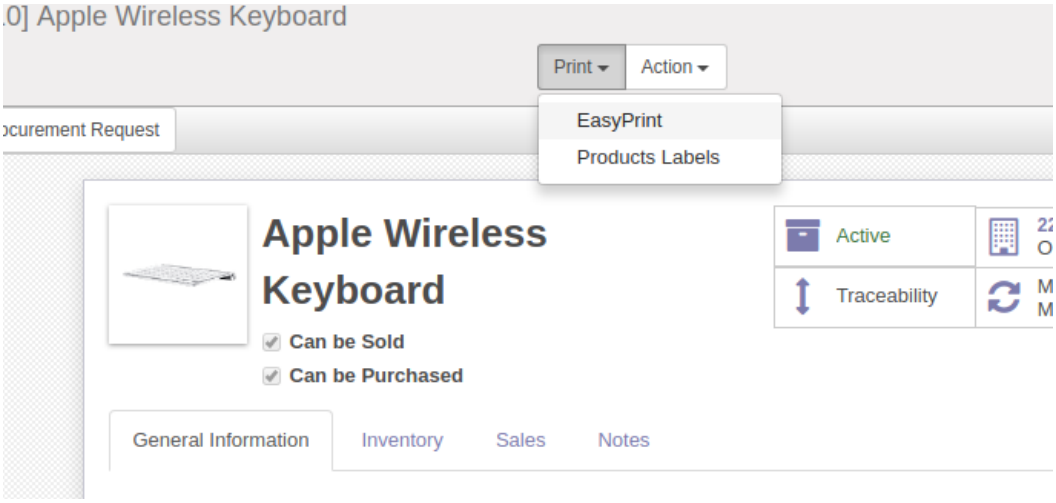
def _get_total_qty(self, ids):
obj = self.env['stock.picking'].browse(ids)
tot = sum(obj.move_lines.mapped('product_uom_qty'))
return str(round(tot,3))
Table can be created using returned value on template mode. So that the called function must be return the data using the following format.
FORMAT: *data = {'header':['col1','col2'],'data':[[112323,435345.5],[23.4,56],[234,345]]}*
Table header values should be described herein. Here are 2 column and 3 raw data.
Syntax: *{function:function_name}. EG*: {function:_get_custom_data}

- Binary and image field: Binary or Image field will be displayed without any size. Syntax: *{image_field:field_name}* . EG: {image_field:image_medium}
  - It allows to display the drawing with the measured sizes. Syntax: *{image_field:field_name:width:height}*. EG: {image_field:image_medium:128:128}

- Drawing to be displayed indicating via URL: In order to display several drawings on the template mode, firstly, URLs to be showed up on Text field should be set using the following format.
  Format:
  [img.test.com/uploads/20160827/c8ad95ad0e7075.png,img.test.com/uploads/order_note/20170829/7d8df4bd453dc2fc10c.jpg]
  After that the text field should be indicated on the template mode.
  Syntax: *{image_field_urls:field_name}*
  EG: {image_field_urls:pic_urls}

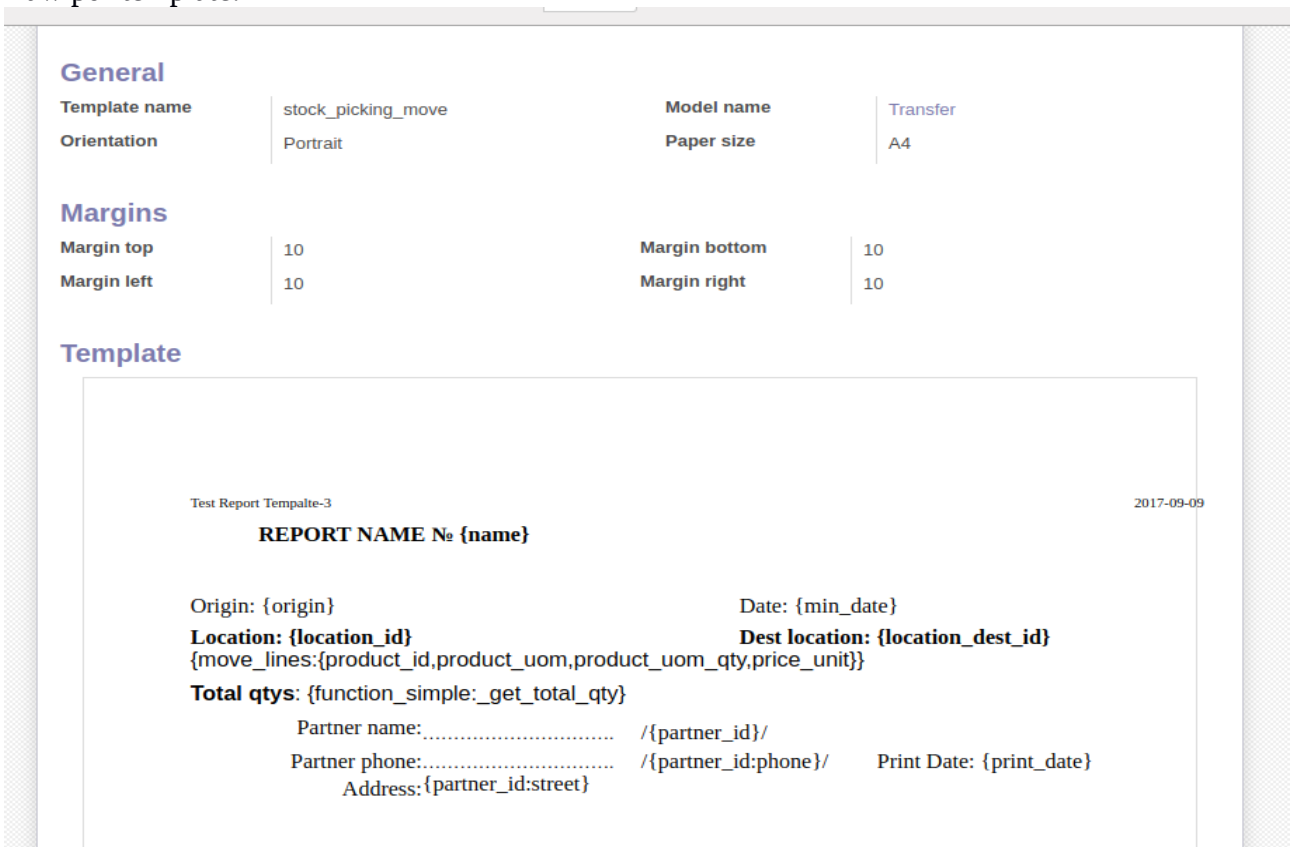
Default Menu

When connecting EasyPrint button of Sidebar menu of all model with default template, it's easy to set template name as "Default".



Appearance

Raw pdf template:



After printing:

Correlation of prepared model and code

There are two steps in order to use the prepared model.

1. PRINT button should be added to XML view. It calls the print function.
2. python function should be written on the model where prepared

EG: PRINT button added to stock.picking:

Step 1

```
<!-- Test picking print -->
<record id="stock_picking_form_inherit" model="ir.ui.view">
  <field name="name">stock.picking.form.inherit</field>
  <field name="model">stock.picking</field>
  <field name="inherit_id" ref="stock.view_picking_form"/>
  <field name="arch" type="xml">
    <xpath expr="//field[@name='name']" position="after">
      <button name="action_to_print" class="oe_highlight"
        string="Print PDF" states="partially_available,assigned,done" type="object"/>
      <button name="action_to_print_move" class="oe_highlight"
        string="Print PDF2" states="confirmed,draft" type="object"
        help="Not ready picking print"/>
    </xpath>
  </field>
</record>
```

Step 2

Call function of the added button should be described on the model.

```
class StockPicking(models.Model):
    _inherit = 'stock.picking'
    # PRINT example2
    @api.multi
    def action_to_print(self):
        model_id = self.env['ir.model'].search([('model','=','stock.picking')], limit=1)
        template = self.env['pdf.template.generator'].search([
            ('model_id','=',model_id.id),
            ('name','=','stock_picking_ready')], limit=1)

        if template:
            res = template.print_template(self.id)
            return res
        else:
            raise UserError(_('Not found print template, Contact to your
administrator!'))
    # PRINT example
    @api.multi
    def action_to_print_move(self):
        model_id = self.env['ir.model'].search([('model','=','stock.picking')], limit=1)
        template = self.env['pdf.template.generator'].search([
```

```
                ('model_id','=',model_id.id),
                ('name','=','stock_picking_move')], limit=1)
        if template:
                res = template.print_template(self.id)
                return res
        else:
                raise UserError(_('Not found print template, Contact to your
administrator!'))
    # FUNCTION_SIMPLE example
    @api.multi
    def _get_total_qty(self, ids):
        obj = self.env['stock.picking'].browse(ids)
        tot = sum(obj.move_lines.mapped('product_uom_qty'))
        return str(round(tot,3))
```

Example code created on stock.picking is located in the module.