2016

odoo

EXPERIENCE

How to create custom accounting reports

Cédric Snauwaert • Senior Software Developer

1. Financial Reports vs Custom Reports
2. How to create your custom report
3. Further customization
4. Q&A

1

Financial Reports Vs Custom Reports

Features Comparison

Features Comparison

Financial

- Very easy to create (XML)
- Ready to use
- Only works with account_move_line
- Only with sums over a period

Custom

- More advanced (Python)
- Possibilities are much more open

Some examples of reports

Financial

- Profit and Loss
- Balance Sheet
- Cash Summary
- Aged Partner Balances
- Most localizations reports

Custom

- Bank Reconciliation
- General Ledger
- Generic Tax Report
- Belgian Partner VAT Listing
- Follow-ups

2

How to create your custom reports

This is the basic interface **to implement.**

```python
from odoo import api
from odoo.tools.translate import _

class my_custom_report(models.AbstractModel):
    _name = "account.my.custom.report"
    _description = "My awesome custom report"

    @api.model
    def get_lines(self, context_id, line_id=None):
        return []

    @api.model
    def get_title(self):
        return _('Awesome report')

    @api.model
    def get_name(self):
        return 'awesome_custom_report'

    @api.model
    def get_report_type(self):
        return self.env['account_report_type'].create()

    @api.model
    def get_template(self):
        return 'account_reports.report_financial'
```

@api.model
get_report_type(self):

This method should return an ***account.report.type*** object.

```
7   class AccountReportType(models.Model):
8       _name = "account.report.type"
9
10      date_range = fields.Boolean('Reports use a date range and not a single date', default=True)
11      comparison = fields.Boolean('Reports allow comparisons', default=True)
12      cash_basis = fields.Boolean('Reports always use cash basis', default=False)
13      analytic = fields.Boolean('Reports enable the analytic filter', default=False)
14      extra_options = fields.Boolean('Display the extra options dropdown (with cash basis and draft entries)', default=True)
```

account.report.type has many fields that modify how mechanics of a specific report work. Many ***account.report.type*** have already been created in the account_reports module but you can always create more combinations or extend the model to add your own type (this will require more work, see further customization)

@api.model
get_lines(self, context_id, line_id=None):

This method should return the list of lines of the report. The line_id parameter is set when unfolding a line in the report, in this case this method should only return a subset of the lines corresponding to line_id and its domain, nothing else.

- The order of the list is the order in which the lines will be displayed in the report
- Each line of the list is a dictionary structure with specific keys

@api.model
get_lines(self, context_id, line_id=None):

Dictionaries for each line can contain the following informations:

- id: id of the line. Referenced for footnote, unfolding, etc.
- name: what will be displayed.
- type: type of the line, this will determine what action is available when clicking on the line. (currently supported: account_id, line, tax_id, unreconciled_aml, bank_statement_id, partner_id, move_line_id, too_many, payment).
- footnotes: {column_number: footnote_id} if there are footnotes for the line.
- unfoldable: True if line can be unfolded.
- unfolded: True if the line has already been unfolded.
- columns: a list of column values.
- level: determine the layout.
- action_id: action that will be triggered when clicking on the line.
- colspan: use this if you want the name to span over many columns.

Also create a report_context.

Report contexts are transient models that are instanciated once per report and per user. You need to create a new context model per custom report. Its interface should be:

```
28  class my_custom_report_context(models.TransientModel):
29      _name = "account.my.custom.report.context"
30      _inherit = "account.report.context.common"
31
32      #If the report is unfoldable, fold_field should contain the name of the field that stores the unfolded lines'
33      fold_field = 'unfolded_lines'
34      unfolded_lines = fields.Many2many('some_model', 'context_to_line', string='Unfolded lines')
35
36      def get_report_obj(self):
37          return self.env['account.my.custom.report']
38
39      def get_columns_names(self):
40          return ['column1', 'column2']
41
42      @api.multi
43      def get_columns_types(self):
44          # Column types can be: text, date or number
45          types = ['number', 'number']
46          return types
```

Last steps

A few last steps before it is in full working order:

- Change the dictionary methods of the account.report.context.common object
  - ***_report_name_to_report_model():***
    Add an item name_of_your_report: name_of_the_model
  - ***_report_model_to_report_context():***
    Add an item name_of_the_model: name_of_the_context_model
- Create the client action and menuitem for your report.

3 Further Customization

Search view

You can customize the search view of your report and adapt it to fit your needs.

- Create a new ***account.report.type***, eg. 'journal_filter'.
- Add fields to your context object that correspond to the filters you want to add.
  Eg. journal_id = fields.Many2one('account.journal')
- Adapt the **get_lines** method in your custom report class
- Modify the ***'accountReports.searchView'*** template by adding a dropdown for your filter.
- Bind events to your dropdown in the ***render_searchview_buttons()*** function of the ***account_report_generic*** widget.

Line type

You can add your own line type and change the way it will appear on the report as well as what actions will be available for that line type. To do that, inherit and modify the ***'account.report_financial_line'*** template.

To trigger an action, use the **'o_account_reports_web_action'** class and then :

- Use **data-res-model** and **data-active-id** to open the form view of the object.
- Use **data-res-model** and **data-action-domain** to open the list view with given domain.
- Use **data-action-name** to trigger that action. You've got the option to add the **data-force-context** attribute to force the current context to be passed on with the action (only relevant if the action goes to another accounting report).
- Use **data-action-id** to trigger that action.

2016

odoo

EXPERIENCE

Thank you.

#odooexperience