场景

1. 页面头部添加 **导入** 和 **下载** 按钮；
2. 点击 **下载模板**，执行指定的python方法，生成excel，作为数据上传的模板；
3. 点击 **导入**，弹出对话框，选择文件，确定导入，随后执行指定的python方法，上传文件中的数据，生成新的数据集；
4. **导入** 和 **下载模板** 按钮都有特定的权限控制；

代码示例：

1、添加上传和下载按钮

在页面顶部添加自定的按钮需要使用qweb模板

在模块根目录下的 **static/src/xml** 文件夹下新建模板xml，并添加到 **__mainifest__.py** 中的 **qweb** 列表中

编辑内容如下：

static/src/xml/pam_import

```
<?xml version="1.0" encoding="utf-8" ?>
<templates id="template" xml:space="preserve">
    <t t-extend="ListView.buttons">
        <t t-jquery="button.o_list_button_add" t-operation="after">
```

```
            <button t-if="widget.model=='pam.failure_type'
                          && widget.options.import_pam_type
                          && widget.options.pam_type_upload"
                    type="button"
                    class="btn btn-primary btn-sm o_pam_type_impo
rt">导入
                <t t-esc="widget.view_id" />
            </button>
            <button t-if="widget.model == 'pam.failure_type'
                          && widget.options.import_pam_failure_t
ype"
                    type="button"
                    class="btn btn-primary btn-sm o_pam_type_dow
nload">下载模板
            </button>
        </t>
    </t>
</templates>
```

说明：

1. <t t-jquery="button.o_list_button_add" t-operation="after">表示，在add按钮（创建按钮有o_list_button_add样式类）后面，如果需要在页面顶部的所有按钮后面追加，则可以使用下面代码；

```
<t t-jquery="div.o_list_buttons" t-operation="append">
```

2. t-if 控制button的显示和隐藏，**&&** 是条件与的关系（上面代码框中打不出来，显示的是&&）；

3. widget.options.import_pam_failure_type 会在js中定义；

2、添加按钮的响应

自定义的按钮的响应需要使用js

在 **static/src/js** 目录下新建js文件，编辑如下：

static/src/js/pam_type_import.js

```
odoo.define('pam_import.tree_pam_type_import_button', function
(require){
"use strict";

var core = require('web.core');
```

```
var Model = require('web.Model');
var ListView = require('web.ListView');
var QWeb = core.qweb;
var data = require('web.data');
var data_manager = require('web.data_manager');
var DataExport = require('web.DataExport');
var formats = require('web.formats');
var common = require('web.list_common');
var Pager = require('web.Pager');
var pyeval = require('web.pyeval');
var session = require('web.session');
var Sidebar = require('web.Sidebar');
var utils = require('web.utils');
var View = require('web.View');
var _t = core._t;
var _lt = core._lt;
var QWeb = core.qweb;
var list_widget_registry = core.list_widget_registry;

ListView.prototype.defaults.import_pam_failure_type = new Model("pam.type").call(
  'check_access_rights', {operation: 'write', raise_exception: false}
);
new Model("pam.type").call('user_has_groups', ['pam.group_type_upload'], {}).then(function(rs){
  ListView.prototype.defaults.pam_type_upload = rs;
});
ListView.include({
  render_buttons: function($node) {
  var self = this;
  this._super($node);
  this.$buttons.find('.o_pam_type_import').click(this.proxy('import_type_file'));
  this.$buttons.find('.o_pam_type_download').click(this.proxy('download_type_template'));
```

```
},
import_type_file: function(){
var self = this;
self.do_action({
name: '导入',
res_model: 'pam_om.type_import_wizard',
type: 'ir.actions.act_window',
views: [[false, 'form']],
target:'new',
});
},
download_type_template: function(){
var self = this;
session.get_file({
url: '/web/type/download/template'
})
}
})
})
```

说明：

1. new Model('pam.type') 会得到 **pam.type** 的模型对象，使用 **call()** 方法可以调用其内部函数。call()方法的三个参数分别是：调用的方法名、参数列表、上下文(默认为空)，并且调用完成之后可以添加回调函数，回调函数的参数是调用方法的返回值；

示例：

```
new Model("pam.type").call('user_has_groups', ['pam.group_type_upload'], {}).then(function(rs){
 ListView.prototype.defaults.pam_type_upload = rs;
});
```

2. this.$buttons.find('.o_pam_type_import').click(this.proxy('import_type_file')); 根据button样式找到响应的button，并添加响应函数；

3. 上例的导入则是根据 **pam_om.type_import_wizard** 模型弹出的视图，需要重新创建数据模型；

4. 上例的导出则是使用 **controller**，这里的url是 **/web/type/download/template**；

3、将按钮的相应文件添加到模块中

使用xml即可

在 **views** 目录下新建xml文件，并添加到**__mainifest__.py** 中的 **data** 列表中

文件编辑如下：

import_template.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<odoo>
 <data>
 <template id="pam_import_type_assets_backend"
 name="service import tree view menu"
 inherit_id="web.assets_backend">
 <xpath expr="." position="inside">
 <script type="text/javascript" src="/pam_import/static/src/js/pam_type_import.js" />
 </xpath>
 </template>
 </data>
</odoo>
```

说明：

1. script的src属性使用的路径要从模块名称开始；

4、添加导入的向导视图

上例中的导入功能，使用的是新的数据模型和视图

在 **wizard** 目录下新建python和xml文件，并添加到**__mainifest__.py** 中的 **data** 列表中

wizard/import_type.py

```
# -*- encoding: utf-8 -*-

from odoo import api, fields, models, _
from cStringIO import StringIO
```

```python
import base64
import xlrd
from odoo.exceptions import ValidationError
from .. import define_data

class FailureTypeExportWizard(models.TransientModel):
    _name = 'pam_om.type_import_wizard'

    to_import = fields.Text(u'需要导入的有效条目')
    filename = fields.Char(u'文件名')
    data = fields.Binary(u'请选择文件上传')

    #导入数据
    def button_next(self):
        try:
            #这里需要获取上传的文件的数据流
            xls_file = StringIO(base64.decodestring(self.data))
        except:
            raise ValidationError(u'请上传模板文件!')
#这里执行数据导入的方法，使用python的xlrd模块即可
        return {'type': 'ir.actions.act_window_close'}
```

wizard/import_type.xml

```xml
<?xml version="1.0" encoding="utf-8" ?>
<odoo>
   <record id="pam_om_import_type_wizard" model="ir.ui.view">
        <field name="name">导入</field>
        <field name="model">pam_om.type_import_wizard</field>
        <field name="arch" type="xml">
            <form string="导出模板">
                <group>
                    <field name="data" filename="filename"/>
                     <field name="filename" invisible="1"/>
                </group>
                <footer>
                    <button name="button_next" type="object"
```

```
                                string="导入" class="oe_highlight"/>
                        <button special="cancel" string="取消"/>
                    </footer>
                </form>
            </field>
        </record>
    </odoo>
```

5、添加导出路由

在 **controllers** 目录下新建python文件，并在目录下的 **__init__.py** 中导入新建的文件

编辑内容如下：

```
# -*- encoding: utf-8 -*-

from odoo import http
from odoo.http import content_disposition, request
from .. import define_data
import StringIO
import xlwt

class Route(http.Controller):
    @http.route(['/web/type/download/template'], type='http', auth='public')
    def download_failure_type_template(self, debug=None, token=None):
        file_name = u'类别模板.xls'
        workbook = xlwt.Workbook()
        workbook.encoding = 'gbk'
        worksheet = workbook.add_sheet(u'类别标准模板')

        column = 0
        for item in define_data.failure_type_items:
            worksheet.write(0, column, item)
            column = column + 1
        string_io = StringIO.StringIO()
        workbook.save(string_io)
```

```
23
24          headers = [
25              ('Content-Type', 'application/vnd.ms-excel;'),
26              ('Content-Disposition',
content_disposition(file_name))
27          ]
28
29          response = request.make_response(string_io.getvalue(),
30                                           headers=headers,
31                                           cookies={'fileToken': t
oken})
32          return response
```

说明：

1. 注意url的匹配；
2. 方法中使用python的xlwt模块导出excel文件；

6、控制权限

使用权限组，控制按钮的显示/隐藏即可

在 **security/security.xml** 中添加权限组

```
1 <record model="ir.module.category" id="type_management_menu_mana
gement">
2     <field name="name">类型管理权限组</field>
3     <field name="description">类型管理权限组</field>
4 </record>
5 <record model="res.groups" id="group_type_upload">
6     <field name="name">类型管理-上传</field>
7     <field name="category_id" ref="pam.type_management_menu_manag
ement"/>
8     <field name="implied_ids" eval="[(4,
ref('base.group_user'))]"/>
9     <field name="comment">类型管理-上传</field>
10 </record>
```

添加权限组的方法具体请参考文章《odoo的权限控制》

数据模型的父类都会有一个 **user_has_groups** 方法以检查当前账户是否在指定的权限组中，我们需要在js中调用此方法，并保存在变量中。最后，xml根据此变量控制是否显示按钮

上例中的js中的代码即是执行此方法，如下：

```
new Model("pam.type").call('user_has_groups', ['pam.group_type_upload'], {}).then(function(rs){
 ListView.prototype.defaults.pam_type_upload = rs;
});
```

这里执行的是 **pam.type** 模型的 **user_has_groups** 方法，传递的参数是 **pam.group_type_upload**，即需要检查的权限组。调用后执行回调函数，得到执行结果保存在 **pam_type_upload** 中

最后，在添加button的xml中根据执行结果显示/隐藏button

```
<button t-if="widget.model=='pam.failure_type'
             && widget.options.import_pam_failure_type
             && widget.options.pam_type_upload"
 type="button" class="btn btn-primary btn-sm o_pam_type_import">导入
 <t t-esc="widget.view_id" />
</button>
```

这里检查 **pam_type_upload** 变量，为True时，显示导入button