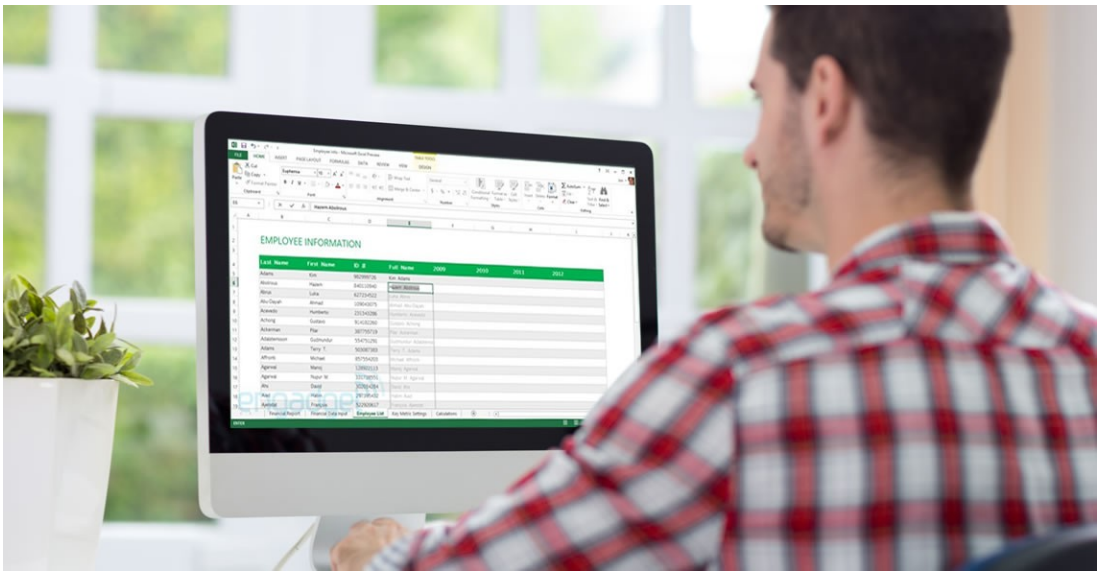
Module EXPORT - Développeur



Résumé

Module qui permet d'exporter une liste Odoo dans un tableur.

Référence Produit : Ecole
Plateforme : Odoo
Version Produit : 11.0.20.02.03

Type de document : Manuel d'utilisation – Utilisateur – Module export
Version du document : 1
Etabli par : Kévin HENNION et MOREAU Clément
Lieu d'utilisation : Parthenay
Nombre de pages : 8

Communauté de Communes de Parthenay-Gâtine

2020

Sommaire

1 Introduction

Le module Export est un module développé majoritairement en python (avec une doctrine Odoo), Les templates sont eux développés en Qweb qui permet de récupérer une liste odoo puis de l'exporter dans un tableur excel ou libre office. Mais d'abord il faut faire passer cette liste du back au front pour afficher un tableau déroulant en fonction de la liste choisi, ce tableau est constitué de checkbox qui permet de cocher les champs que l'on veut ou pas. Ensuite une fenêtre modale s'ouvrira pour donner d'abord un aperçu du resultat avant de l'exporter : il faut choisir le nom du fichier à générer et l'exportation est à réaliser.

2 Arborescence du dossier

Ci contre l'arborescence du dossier export_view_parthenay

n'oubliez pas d'importer les fichiers voulus dans les

__init__.py, de bien remplir le fichier __manifest__.py

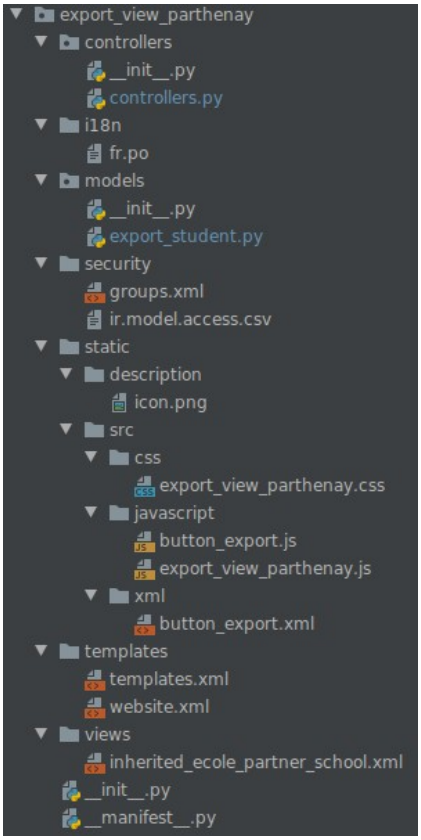
et bien mettre dans 'data' (dans le fichier __manifest__.py) les fichiers xml que vous charger

par exemple.

L'arborescence est plutôt basique, tous les fichiers js

sont rangés ensemble, et ceci pour tous les fichiers css,

controller , model etc...



3 Export_fichier

Dans export_view_parthenay/controllers/controllers.py :

Le contrôleur est constitué de route comme ci dessous. Chaque route correspond à une page, mon @http.route correspond à http://localhost:8011 puis on y ajoute ce qui est entre parenthèse ('/export_fichier/<data_back>') sachant que data_back correspond à un tuple qui est encore en brut dans le code : ecole.partner.school,5,14,7,11,13,18,16,17. Les chiffres sont les id des élèves que je veux afficher pour le test.

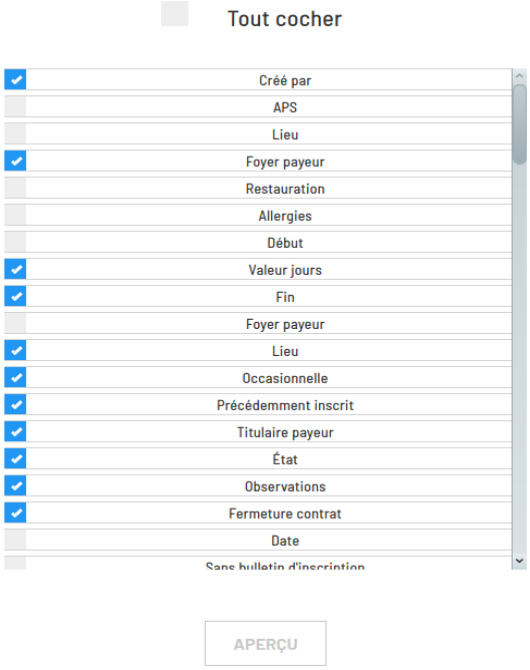
Alors pour accéder à cette page nous devons mettre dans l'url : http://localhost:8011/export_fichier/ecole.partner.school,5,14,7,11,13,18,16,17

```python
@http.route('/export_fichier/<data_back>', auth='user', website=True, csrf=False, type='http', method=['POST'])
def indexExport(self, data_back, **post):
    data_back = data_back.split(",")

    # retourne des éléments au template (comme un select en sql)
    fields_model = http.request.env['ir.model.fields'].search([('model', '=', data_back[0]),
                                                               ('store', '=', True),
                                                               ('name', 'not in', ['create_date', 'write_date'])])

    return http.request.render('export_view_parthenay.export', {
        'fields_model': fields_model,
        'data_back': data_back,
    })
```

Aperçu code dans export_view_parthenay/controllers/controllers.py (ligne 17 à 29)



Aperçu http://localhost:8011/export_fichier/ecole.partner.school,5,14,7,11,13,18,16,17

export_fichier correspond à la première page en front du module.

data_back est un tuple qui récupère les id des élèves qui sont pour l'instant encore en brut.

fields_models récupère le nom des colonnes.

ecole_partner_school est la table qui contient tous les élèves.

ir_model.fields regroupe tous les champs de toutes les tables.

Dans export_view_parthenay/templates/template.xml :

```
<form method="POST" id="form_data" name="form_ecole_partner_school" onsubmit="return result_requests_export()">
    <input type="hidden" name="csrf_token" t-att-value="request.csrf_token()"/>
    <div class="all_check_div">
        <label class="all_check"> Tout cocher
            <input type="checkbox" name="checkAll" id="all" onclick="check()"/>
            <span class="checkmark2"></span>
        </label>
    </div>
    <div class="list_checkbox_ecole_partner_school">
        <t t-foreach="fields_model" t-as="rec">
            <label class="label_ecole_partner_school">
                <input type="checkbox" class="checkbox_fields" t-att-name="rec.field_description" t-att-value="rec.field_description"/>
                <span class="checkmark"/>
                <t t-esc="rec.field_description"/>
            </label>
        </t>
    </div>
    <input type="hidden" id="data_back" name="data_back" t-att-value="data_back"/>
    <div class="btn_form_ecole_partner_school">
        <input value="Aperçu" type="submit" class="btn btn-outline-primary" name="btn_form_ecole_partner_school" data-toggle="modal"
               data-target="#myModalResult"/>
    </div>
</form>
```

Aperçu code dans export_view_parthenay/templates/template.xml(ligne 7 à 29)

Le premier template (id='export' commence ligne 3 et fini ligne 50) dans template.xml correspond au visuel de cette page « export_fichier » il est composé d'un formulaire qui est composé de checkbox qui affiche field_description .

ici on boucle sur fields_model et on affiche field_description

field_description est le nom des champs en Français de la table ir.models.fields

Dans le label de class 'label_ecole_partner_school', le input permet de savoir qu'une checkbox correspond à quel valeur de field_description, le span, c'est la case checkbox et le t-esc, c'est juste l'affichage de field_description.

La div de class: 'all_check_div' correspond à la checkbox « tout cocher » qui est faite pour cocher ou décocher toutes les checkbox. Elle appelle la fonction « check() » qui est une fonction javascript située dans export_view_parthenay/static/javascript/export_view_parthenay.js

le formulaire retourne « result_requests_export() » qui est aussi dans export_view_parthenay/static/javascript/export_view_parthenay.js

```
document.getElementById("request_export").src = '/tableau_export_fichier/'+checkbox_field_list.toString()+'/'+data_back.value.toString();
```

checkbox_field_list.toString() et data_back.value.toString() se retrouvent ici :

```
@http.route('/tableau_export_fichier/<fields_checked>/<data_back>', auth='user', website=True, type='http',
            csrf=False, method=['POST'])
def tableauExport(self, fields_checked, data_back, **post):
```

fields_checked récupère les checkboxs qui ont été cochées

4 tableau_export_fichier

Ici nous affichons le tableau après avoir cliqué sur « Aperçu » :

Nous refaisons le tableau grâce au fields_checked (checkbox cochées) et le data_back

vals n'est pas une liste mais un dictionnaire que l'on affiche dans le template pour faire le tableau. Il contient les champs selectionnés (nom des colonnes).

```
<table class="table table-striped" >
        <thead>
            <tr>
                <t t-foreach="vals" t-as="item">
                    <th><t t-esc="item"/></th>
                </t>
            </tr>
        </thead>
```

Dans le template (id="modal" commence ligne 52 et finit ligne 98) nous affichons dans le tableau : vals

dans le thead on affiche les checkboxs cochées précédemment

```
<t t-set="i" t-value="0"/>
<t t-foreach="ids" t-as="record">
    <tr>
      <t t-foreach="vals" t-as="item">
          <t t-if="len(item_value) > i">
              <t t-if="item_value[i]">
                  <td><t t-esc="item_value[i]"/></td>
              </t>
              <t t-else="">
                  <td>-</td>
              </t>
          </t>
          <t t-else="">
              <td>-</td>
          </t>
      </t>
    </tr>
<t t-set="i" t-value="i+1"/>
</t>
```

Ids ce sont les id de la table ecole partner school (en fonction de ce que l'utilisateur a coché)

ici dans le tbody on boucle pour afficher le contenu du tableau

5 telechargement_fichier

Telechargement fichier est de la ligne 104 à 165 dans export_view_parthenay/controllers/controllers.py

Grâce à la méthode post, nous récupérons les contenus du tableau (vals) mais en string, donc nous devons le reconstruire en format dictionnaire :

```
# On reconstitue le dictionnaire qui était en format str
data_result = ast.literal_eval(post.get('vals', False))
```

name_file est le nom du fichier entré par l'utilisateur et on lui rajoute le « .xls»

puis de la ligne 115 à 138, c'est le style du tableur excel, c'est à dire , le design du tableur.

Ensuite de la ligne 140 à 150 le traitements des données récupérées, donc comment elles seront disposées dans le tableur, ligne, colonne etc...

```
# Traitement des données pour export
row = 0
column = 0
if row == 0:
    for key, values in data_result.items():
        worksheet.write(row, column, key, for_left)
        i = 1
        for value in values:
            worksheet.write(row + i, column, value, for_left_not_bold)
            i += 1
        column += 1
```

Cette partie de code (l-152 à 165) sert à exporter le fichier :

il sera téléchargé en fonction de votre navigateur et aussi stocké dans la table export.student.

```
fp = io.BytesIO()
# Sauvegarde des fichiers apres écriture (write)
workbook.save(fp)
# Stockage sous forme binaire du fichier dans la base Odoo (table export.student)
export_id = http.request.env['export.student'].create(
    {'excel_file': base64.encodestring(fp.getvalue()),
     'file_name': filename})
fp.close()

url_file = '/web/binary/download_document_partner?model=export.student&field=excel_file&id=' + str(
    export_id.id) + '&filename=' + filename

return werkzeug.utils.redirect(url_file)
```

6 export_student.py

```python
@api.model
def get_active_records(self):
    # records = self.student_ids
    # active_ids = self.ids
    # active_ids_two = self.env.context.get('active_ids', [])
    # print(records)
    # print(active_ids)
    # print(active_ids_two)
    return [['ecole.partner.school'], [5,14,7,11,13,18,16,17]]
```

ecole_partner_school est la table qui contient tous les élèves.

5,14,7,11,13,18,16,17 ce sont les id des élèves. Pour l'instant ils sont en brut car je n'ai pas eu le temps de trouver la solution durant mon stage.