
om5 Documentation

Version 1.2.0a1

om5

août 07, 2024

Table des matières

1	Installateur	3
1.1	Pré requis d'installation :	3
1.2	Installateur :	5
2	Développement Rapide	9
2.1	Saisir table et génération :	9
2.2	Saisir des champs dans la table :	12
2.3	Supprimer les contraintes de clé secondaire :	15
2.4	Saisir des actions :	16
2.5	Saisir des triggers :	18
2.6	Saisir les procédures :	20
2.7	Saisir des vues :	23
2.8	Exemple : la gestion de ma bibliothèque :	27
3	Technique	37
3.1	Le framework openMairie :	37
3.2	La surcharge om_gen_plus.class.php :	38
4	Bibliographie	41
5	Contributeurs	43

Note : Cette création est mise à disposition selon le Contrat Paternité-Partage des Conditions Initiales à l'Identique 2.0 France disponible en ligne <http://creativecommons.org/licenses/by-sa/2.0/fr/> ou par courrier postal à Creative Commons, 171 Second Street, Suite 300, San Francisco, California 94105, USA.

om5_no_code a pour objet de créer des applications sans avoir besoin d'écrire une seule ligne de code. En effet, om5_no_code est une sur-couche du framework openMairie et propose de générer (sans compétences en langage informatique) une application qui aurait autrefois nécessité l'intervention d'un développeur spécialisé. Autrement dit, de créer du code, sans coder !

om5_no_code propose au niveau d'abstraction de l'analyse, la génération de code (formulaire, édition) et l'utilisation des ressources de la base de données postgresql (triggers, procédures, vues).

Note : La partie « triggers, procédures, vues » est en cours de développement.

L'interface graphique augmente la productivité des développeurs mais la contrepartie de cet avantage est la généralité et avec om5_no_code, une interface générique ; mais il est possible de revenir sur l'utilisation du framework openMairie. (surcharge des classes métier générées, css, javascript ...).

Enfin, l'utilisation des techniques déclaratives visuelles au lieu de la programmation permet aux experts métier de diriger ou de participer à la fourniture de la solution.

Ce document a pour but de guider les utilisateurs et les développeurs dans la prise en main du projet om5-no-code.

Il est composé de 2 modules :

- un installateur du framework,
- un outil de développement rapide qui permet de créer les tables et de générer des formulaires et des éditions.

La partie installateur décrit le fonctionnement du module d'installation du framework openMairie.

La partie developpement rapide, décrit le fonctionnement de ce module autour d'un exemple de bibliothèque.

La partie technique propose de décrire l'intégration du projet om5_rad dans le framework openMairie.

Bonne lecture et n'hésitez pas à nous faire part de vos remarques à l'adresse suivante : contact@openmairie.org !

Les sources et le téléchargement du projet sont sur la forge de l'ADULLACT au lien suivant :

<http://adullact.net/projects/om5-no-code/>

Ce chapitre décrit l'installateur du projet.

1.1 Pré requis d'installation :

Le framework openMairie 4.10.0 nécessite l'installation

- du serveur web APACHE,
- du module php <= 8.0
- de la base de données POSTGRES

Sources : (debian 9 / juin 2020)

https://wiki.arles-linux.org/doku.php?id=install_debian

1.1.1 Installation des composants sur debian 11

installation serveur apache et module php

```
apt install -y apache2
apt install -y php
apt install -y php-pgsql
apt install -y php-mbstring
apt install -y php-xml
# redémarrer apache
systemctl restart apache2
```

installation de la base de données postgresql et de la cartouche géographique postgis

```
apt install -y postgresql
apt install -y postgresql-contrib
apt install -y postgis
```

création d'un utilisateur deb pour postgres (avec droits de créer une base de données, schéma, tables ...)

Sources :

https://wiki.arles-linux.org/doku.php?id=securite_postgres

```
su postgres
createuser -P deb -U postgres
    Saisir le mot de passe pour le nouveau rôle : deb
    Le saisir de nouveau : deb
psql -U postgres
postgres=#
    alter role deb with superuser;
    alter role deb with createdb;
    alter role deb with createrole;
```

créer une base de données om5-no-code

```
createdb om5-no-code
```

télécharger le fichier sur la forge de l'adullact dans /var/www/html : décompresser om5-no-code_[version].zip

Voir explication sur openmairie.org

<https://openmairie.readthedocs.io/projects/omframework/fr/4.9/installation/index.html#telecharger-l-archive-zip>

créer le répertoire var :

```
/var/www/html/om5-no-code $ mkdir var
```

www-data (utilisateur apache) doit avoir les droits d'écriture sur dyn, gen et var

```
/var/www/html/om5-no-code $ chown -R www-data:www-data var
/var/www/html/om5-no-code $ chown -R www-data:www-data dyn
/var/www/html/om5-no-code $ chown -R www-data:www-data gen
```

dans le navigateur, aller sur le répertoire de l'application app/om_setup_config.php ou directement dans l'application /var/www/html/om5-no-code

dans la configuration standard, le database.inc.php se configure par défaut de la manière suivante : # base = om5_rad ; schema : om5 ; user = deb ; password = deb ;

1.1.2 Cas de la debian 12 et de compatibilité 8.0

Le framework 4.10.0 n'est pas compatible avec la version php 8.2 fournie par debian 12.

Il faut mettre php 7.4 ou 8.0 au lieu de la version de base 8.2

Sources :

<https://www.linuxtricks.fr/wiki/debian-installer-une-version-plus-recente-de-php>

<https://sys-admin.fr/installation-php-7-4-sur-debian/>

<https://www.interserver.net/tips/kb/change-php-version-apache-ubuntu/>

Pour installer php 7.4, il faut récupérer les packages sur <https://packages.sury.org/php/>

#liste des clés dans usr/share/keyrings

```
apt-key list
```

ajouter la clé deb.sury.org-php.jpg

```
wget https://packages.sury.org/php/apt.gpg -O /usr/share/keyrings/deb.sury.org-php.gpg

# ajout de la ligne "deb [signed-by=/usr/share/keyrings/deb.sury.org-php.gpg]
# https://packages.sury.org/php/ bookworm main" dans /etc/apt/sources.list.d/php-sury.
↳list

echo "deb [signed-by=/usr/share/keyrings/deb.sury.org-php.gpg] https://packages.sury.
↳org/php/ $(lsb_release -sc) main" > /etc/apt/sources.list.d/php-sury.list
```

installer la version de php 7.4

```
apt update

# install php7.4 -> création du repertoire etc/php/7.4

apt install php7.4
apt install php7.4-pgsql
apt install php7.4-mbstring
apt install php7.4-xml
```

changer le module php dans apache

```
# desactiver la version 8.2 de php
sudo a2dismod php8.2

# activer la version 7.4 de php
sudo a2enmod php7.4
```

1.2 Installateur :

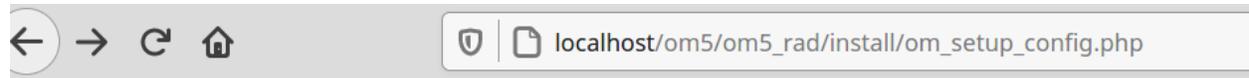
note :

Attention, la procédure d'installation détruit le schéma existant d'om5.

Lors du lancement dans le répertoire du framework , si le fichier dyn/database.inc.php n'existe pas, il est lancé l'installateur qui est dans install/om_setup_config.php

Dans tout les cas, il faut que database.inc.php soit inexistant sinon le framework démarre sur les paramètres de ce fichier.

Démarrer en cliquant sur le lien : « allez c'est parti ! »



Bienvenue sur l'installateur d'openFramework 4.10.0

Avant de nous lancer, nous avons besoin de certaines informations sur votre base de données. Il va vous falloir réunir les informations suivantes pour continuer.

- Nom de la base de données
- Nom du schéma de données
- Nom d'utilisateur Postgres
- Mot de passe de l'utilisateur
- Hôte de base de données
- Port de la base de données (par défaut 5432)

Nous allons utiliser ces informations pour créer le fichier dans dyn : database.inc.php.

Si pour une raison ou pour une autre la création automatique du fichier ne fonctionne pas, ne vous inquiétez pas. Sa seule action est d'ajouter les informations de la base de données dans un fichier de configuration. Vous pouvez aussi simplement ouvrir dyn/database.inc-sample.php dans un éditeur de texte, y remplir vos informations et l'enregistrer sous le nom de database.inc.php. Vous devriez normalement avoir reçu ces informations de la part de votre hébergeur. Si vous ne les avez pas, il vous faudra contacter votre hébergeur afin de continuer.

Si vous avez besoin d'aide, contactez le [forum](#)

vous avez des explications complémentaires au lien suivant [documentation du projet om5_rad](#)

[Cliquez sur ce lien pour configurer le fichier database.inc.php](#)

1.2.1 Création du database.inc.php :

La première étape consiste à créer le lien entre l'application et postgres en créant le fichier dyn/database.inc.php

Vous devez saisir ci-dessous les détails de connexion à votre base de données. Si vous ne les connaissez pas, contactez votre hébergeur

Nom de la base de données	<input type="text" value="om5_rad"/>	Le nom de la base de données avec laquelle vous souhaitez utiliser le framework
Nom du schéma de la base de données	<input type="text" value="om5"/>	Le nom du schéma avec lequel vous souhaitez utiliser le framework
Identifiant	<input type="text" value="postgres"/>	Nom d'utilisateur postgresql
Mot de passe	<input type="text" value="postgres"/>	Votre mot de passe de base de données
Adresse de la base de données	<input type="text" value="localhost"/>	Si localhost ne fonctionne pas, demandez cette information à l'hébergeur de votre site.
Port du serveur de base de données	<input type="text" value="5432"/>	Si 5432 ne fonctionne pas, demandez cette information à l'hébergeur de votre site.

Si vous avez les droits d'écriture sur le répertoire /dyn, le fichier database.inc.php est automatiquement créé. Sinon copier dans un fichier dyn/database.inc.php, le contenu du texte affiché :

la connexion est ok
 Vous avez accès en écriture et le fichier a été copié en repertoire dyn avec les données suivantes

```
<?php
$conn[1] = array(
    'om5_rad', // Titre
    'pgsql', // Type de base
    'pgsql', // Type de base
    'postgres',
    'postgres', // Mot de passe
    '', // Protocole de connexion
    'localhost', // Nom d hote
    '5432', // Port du serveur
    '', // Socket
    'om5_rad', // Nom de la base
    'AAAA-MM-JJ', // Format de la date
    'om5', // Nom du schéma
    '', // Préfixe
    null // Paramétrage pour l'annuaire LDAP
);
```

om5_rad	om5	postgres	postgres	localhost	5432
---------	-----	----------	----------	-----------	------

Créer le schéma des données pour le framework

vous pouvez aussi utiliser la procédure data/nsal/install.sql

1.2.2 Installation du schéma du framework :

Les scripts nécessaires au framawork sont lancés :

```
SET client_encoding = 'UTF8'; executé
SET search_path = om5, public, pg_catalog; executé
CREATE EXTENSION IF NOT EXISTS postgis; executé
DROP SCHEMA IF EXISTS om5 CASCADE; executé
CREATE SCHEMA om5 executé
var/www/html/om5/om5_rad/core/data/pgsql/init.sql
279 lignes traitées
var/www/html/om5/om5_rad/core/data/pgsql/init_permissions.sql
121 lignes traitées
var/www/html/om5/om5_rad/core/data/pgsql/init_parametrage.sql
3 lignes traitées
Ajout des vues et des droits du projet om5_rad
CREATE OR REPLACE VIEW om5.om_tables AS SELECT table_name, table_schema, table_type FROM information_schema.tables where table_type = 'BASE TABLE' and table_schema = 'om5' executé
CREATE OR REPLACE VIEW om5.om_champs AS SELECT concat(table_name, '.', column_name) as column_name, table_name, table_schema, data_type, is_nullable, character_maximum_length FROM
information_schema.columns where table_schema = 'om5' executé
CREATE OR REPLACE VIEW om_constraints AS SELECT tc.constraint_name, tc.table_name, kcu.column_name, ccu.table_name AS foreign_table_name, ccu.column_name AS foreign_column_name FROM
information_schema.table_constraints AS tc JOIN information_schema.key_column_usage AS kcu USING (constraint_schema, constraint_name) JOIN information_schema.constraint_column_usage AS ccu
USING (constraint_schema, constraint_name) WHERE constraint_type = 'FOREIGN KEY' AND tc.table_schema = 'om5' executé
INSERT INTO om5.om_droit (om_droit, libelle, om_profil) VALUES (nextval('om5.om_droit_seq'), 'om_tables', 1) executé
INSERT INTO om5.om_droit (om_droit, libelle, om_profil) VALUES (nextval('om5.om_droit_seq'), 'om_constraints', 1) executé
terminé
```

[Cliquez sur ce lien pour lancer le framework](#)

En cliquant sur le lien affiché, vous lancez le framework avec le login (admin) et le mot de passe (admin).

Développement Rapide

Ce chapitre décrit le module de développement rapide du projet om5.

Le premier objectif est de créer les tables avec les champs et les contraintes de clé secondaire, et de générer les listes et les formulaires.

Le deuxième objectif est de créer des traitements avec les triggers ; les procédures et les vues.

2.1 Saisir table et génération :

Il est possible de lister les tables dans le menu om5-no-code -> option Gestion de Tables

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets

Ce listing décrit les tables existantes pour votre application

Table								
1 - 3 enregistrement(s) sur 3				Tous		Recherche		
+	table	libelle	col1	col2	col3	menu	nb_col	recherche
	livre	Ma bibliothèque	Titre	Résumé	colonne 3	1	2	1
	emprunteur	Mes Amis	Etat Civil	colonne 2	colonne 3	1	1	1
	pret	prêt	livres ou amis	Retour	colonne 3	0	2	0

Il est possible de saisir, modifier et supprimer les tables dans le formulaire des tables :

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Livre Om5

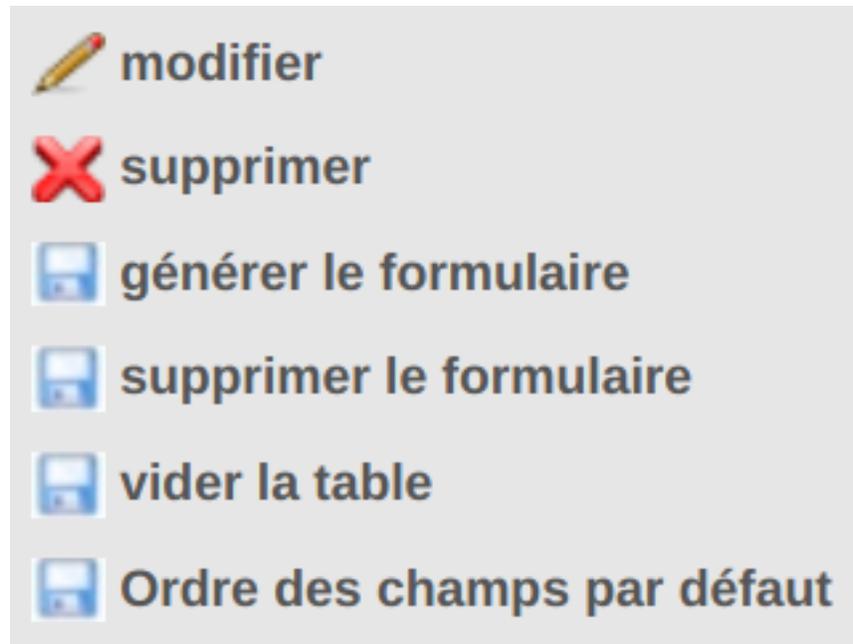
Table Formulaire Action clé secondaire Trigger postgresql

Modifier  Retour

Nom de la table * livre
libellé de la table livre
affichage en menu afficher en menu ▾
nombre de colonnes 2 colonnes ▾
libellé colonne 1 Etat Civil
libellé colonne 2 Activités
moteur de recherche avec ▾

Modifier  Retour

Les actions suivantes sont possibles :



2.1.1 Action ajouter modifier et supprimer :

Lors de l'action ajouter :

- il est vérifié que la cohérence « postgresql » du nom de table ne soit pas vide et qu'il ne soit pas existant
- le nom d'origine (non corrigé) est dans le champs libellé
- il est possible de saisir l'affichage en menu, le nombre de colonne du formulaire et le nom de chaque colonne
- la table est créée
- la clé primaire est créé automatiquement, elle est de type integer et elle n'accepte pas de null
- un droit est créé sur la table dans om_droit avec le profil utilisateur
- la table est insérée dans le menu option application si l'affichage est ok avec le libellé.

Note : Le nom de la table est automatiquement corrigé car il doit correspondre aux principes de nommage des tables par postgresql : pas de caractères spéciaux ou blanc, pas de majuscule, commence par une lettre ou underscore , ne

contient que des lettres ou des chiffres ou underscore. Le nom d'origine est stocké en libellé et il est modifiable.

Note : Les paramètres sont contenus dans le champs parametres de la table om_tables_parametre au format json

Lors de l'action modifier :

- il est possible de modifier le champs libellé
- il est possible de saisir l'affichage en menu, le nombre de colonne du formulaire et le nom de chaque colonne

Lors de l'action supprimer :

- il est vérifié que la table soit vide
- il est vérifié que le formulaire n'existe plus
- la table est supprimée
- ensuite la sequence est détruite
- puis les paramètres d'om_tables_parametres et om_forms sont détruits
- et le ou les droits sur la table sont détruits

2.1.2 Action générer le formulaire :

- la liste et le formulaire est générée
- la séquence est créée
- les paramètres des champs (om_forms) et de la table (om_tables_parametre) sont pris en compte.
- il est pris en compte les actions supplémentaires du sous formulaire om_actions

Le message suivant est affiché :

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Emprunteur Om5

Table Formulaire Action clé secondaire Trigger pgsql

Table : emprunteur

Génération de ../gen/sql/pgsql/emprunteur.inc.php
Génération de ../gen/obj/emprunteur.class.php

Retour

Nom de la table	emprunteur	modifier supprimer générer le formulaire
Nom du schema	om5	
Type de table	BASE TABLE	
libellé de la table	Mes Amis	

2.1.3 Action supprimer le formulaire :

la liste et le formulaire sont supprimés

2.1.4 Action vider la table :

cette action vide la table des enregistrements.

2.1.5 Action ordre des champs par défaut

Cette action met les champs dans l'ordre de saisie pour la colonne.

2.2 Saisir des champs dans la table :

Il est possible de lister les champs dans le menu om5-no-code -> option Gestion de Tables sous formulaire de la table :

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Pret Om5

Table Formulaire Action clé secondaire Trigger pgsq

1 - 4 enregistrement(s) sur 4

+	champ	table	type	clé secondaire	obligatoire	libelle	type	bloc	no	calcul	liste	no_l	lib_l
	pret.date_de_pret	pret	date		Oui	date de prêt	C1		3		1	3	date de prêt
	pret.emprunteur	pret	integer	pret_emprunteur_fkey	Oui	emprunteur	C1		1		1	1	emprunteur
	pret.livre	pret	integer	pret_livre_fkey	Oui	livre	C1		2		1	2	livre
	pret.retour	pret	boolean		Non	retour	C1		4		1	4	retour

Il est possible de saisir, modifier et supprimer les champs dans le formulaire des champs :

Administration ➔ Om_tables ➔ Livre Om5

om_tables om_champs om_contraintes

administration ➔ om_champs ➔ livre.fichier livre

Modifier Retour

Nom du champs * livre.fichier

Type de champs *

Obligatoire *

column_default

libelle

Type om5

Modifier Retour

Le nom du champ doit être unique pour la table et il doit être rempli.

Note : Le nom du champs est automatiquement corrigé car il doit correspondre aux principes de nommage des champs par postgresql : pas de caractères spéciaux ou blanc, pas de majuscule, commence par une lettre ou underscore , ne contient que des lettres ou des chiffres ou underscore. Le nom d'origine est stocké en libellé et il est modifiable.

Les paramètres sont contenus dans le champ parametres de la table om_forms au format json.

Le data_type de champ peut être :

- character varying
- entier (integer)
- décimal (numéric)
- booleen (boolean)
- date
- texte (text)
- clé secondaire

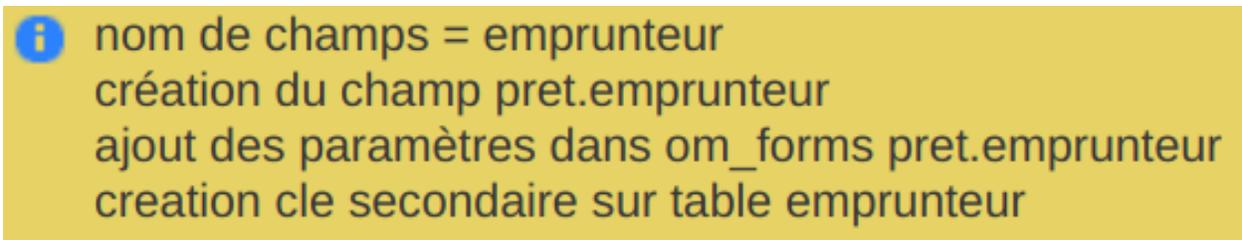
Le type (om) est par défaut celui défini dans om_champs et il est visible :

- Si il est text, il peut devenir html (éditeur html)
- si il est character varying, il peut devenir un champs fichier (file) pour télécharger des fichiers
- il peut être non visible à condition d'être non obligatoire

2.2.1 data_type clé secondaire

Le data type « clé secondaire » propose de choisir la table sur lequel sera créer la clé comme dans l'image suivante :

le message suivant s'affiche :



Note : Le cas « clé secondaire » est particulier et ne peut être utilisé que si une table portant le même nom que le champ existe.

Le champ est alors de type integer.

En cas de suppression, la clé secondaire est automatiquement supprimée.

2.2.2 Champ obligatoire

Le champ peut être obligatoire (n'accepte pas les null) ou pas (accepte les nulls)

Attention, un champ obligatoire ne doit pas être caché

2.2.3 Action sous état pour champ clé secondaire

Il est possible de générer un sous formulaire avec l'action « générer un sous formulaire ». Il faut rendre le sous état actif et l'insérer dans l'état principal.

2.2.4 Position et colonne

En création la position est automatiquement créée dans le bloc C1 (colonne 1) pour le formulaire et le libellé est créée sur la base du nom du champs

La position en liste se fait de la même manière.

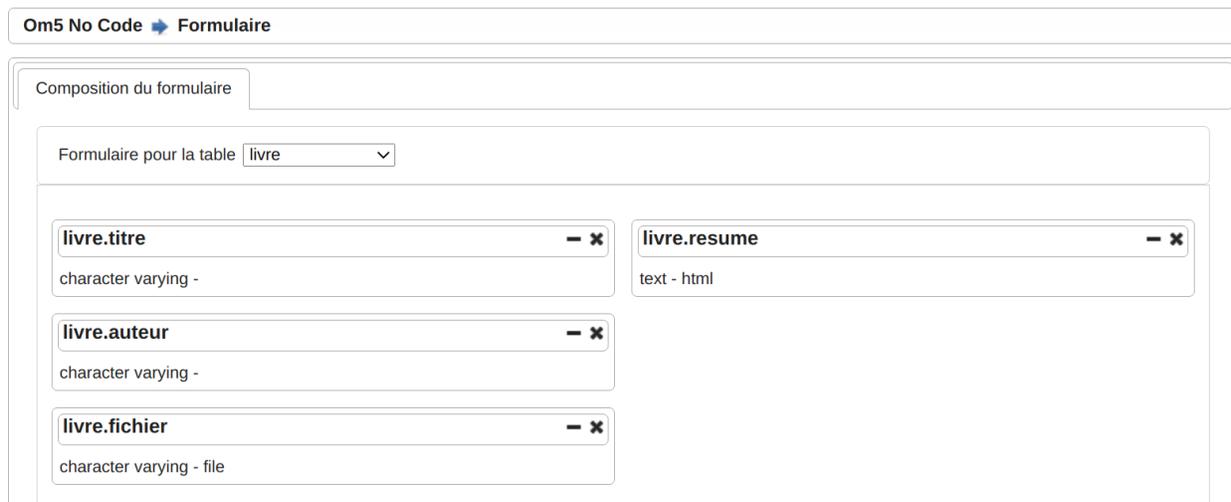
Il est possible de modifier les positions et colonnes dans les formulaires avec l'option du menu om5-no-code : composition formulaires

Il est possible de modifier les positions les listes avec l'option du menu om5-no-code : composition listes

2.2.5 Composition du formulaire :

L'option « composition formulaires » du menu om5-no-code permet de composer le formulaire en disposant les champs dans l'ordre et les colonnes.

La colonne (bloc) et la colonne position est modifiable par glisser/copier.



La suppression de champs se fait en cliquant sur la croix en haut à droite du champs.

Note : Dans la composition du fomulaire, la suppression d'un champs, rend le champs non visble avec un libellé vide.

Note : Il faut générer pour prendre en compte toute modification des champs

2.2.6 Composition de liste :

L'option « composition listes » du menu om5-no-code permet de composer les colonnes de la liste en disposant dans l'ordre les colonnes.

Om5 No Code ➔ Liste

Composition de la liste

Formulaire pour la liste emprunteur ▼

emprunteur.nom	- x
character varying -	
emprunteur.prenom	- x
character varying -	
emprunteur.date_de_naissance	- x
date -	
emprunteur.actif	- x
boolean -	

Note : Il faut générer pour prendre en compte toute modification des champs

2.3 Supprimer les contraintes de clé secondaire :

Il est possible de lister les contraintes de clé secondaire dans le menu om5-no-code -> option Gestion de Tables sous formulaire de om_table :

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Pret Om5

Table Formulaire Action clé secondaire Trigger pgsq

1 - 2 enregistrement(s) sur 2

	▶ contrainte	▶ schema	▶ table	▶ champ	▶ foreign table	▶ foreign champ
	pret_emprunteur_fkey	om5	pret	emprunteur	emprunteur	emprunteur
	pret_livre_fkey	om5	pret	livre	livre	livre

Il est possible de supprimer les contraintes de clé secondaires dans le formulaire des contraintes :

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Pret Om5

Table Formulaire Action clé secondaire Trigger pgsq|

administration ➔ om_contraintes ➔ pret_emprunteur_fkey pret

[Retour](#)

Contrainte	pret_emprunteur_fkey	
Table	pret	
Champs	emprunteur	
Foreign table	emprunteur	
Foreign champ	emprunteur	

[Retour](#)

2.4 Saisir des actions :

Il est possible de lister les procédures dans le menu om-no-code -> option Gestion de Tables sous formulaire de om_actions :

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Livre Om5

Table Formulaire Action clé secondaire Trigger pgsq|

1 - 1 enregistrement(s) sur 1

+	om_actions	libellé	module	ordre	titre	table_name
		1 livre	edition		50 Livre en prêt	Ma bibliothèque

Il est possible de modifier et supprimer les actions dans le sous formulaire actions :

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Livre Om5

Table Formulaire Action clé secondaire Trigger pgsq|

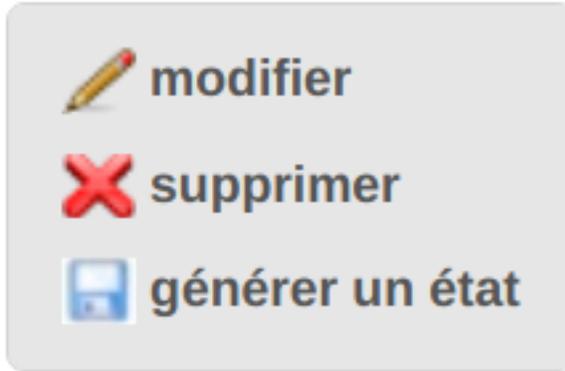
administration ➔ om_actions

Ajouter [Retour](#)

libellé *	<input type="text" value="livre"/>
module *	<input type="text" value="edition"/>
titre	<input type="text" value="Livre en prêt"/>
ordre	<input type="text" value="50"/>
table_name	Ma bibliothèque

Ajouter [Retour](#)

Les actions dont les suivantes :



L'action « générer un état » est accessible pour les actions du module « édition ».

2.4.1 Générer un état :

Cette action permet de générer un état dans la table om_état et une requête dans la table om_requête.

Le message suivant est affiché :



Note : L'état généré est non actif. Il faut aller sur le menu « paramètre » option état et paramétrer l'état en mode actif. La prise en compte de l'action se fait par la génération de la table (action générer d'om_tables)

Il faut donc aller dans le menu paramètres dans l'option Etat :

état	id	libellé	actif	niveau
1	livre_1	livre_1 no code le 14/05/2024	Non	1

et modifier l'état correspondant, il porte comme identifiant le nom de la table avec le numéro de l'action :

Ne pas oublier de cocher actif pour que l'état puisse être utilisé dans l'action d'édition.

En générant dans om_tables (action générer le formulaire), l'action est accessible dans le formulaire comme dans l'image suivante (action : livre en prêt :

2.5 Saisir des triggers :

Note : Attention ce formulaire est en cours de développement.

Il est possible de lister les triggers dans le menu om-no-code -> option om_tables - onglet trigger

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Essai Om5

Table Formulaire Action clé secondaire Trigger postgresql

1 - 0 enregistrement(s) sur 0

+ ▶ nom ▶ evenement ▶ table ▶ ordre ▶ condition ▶ état ▶ orientation ▶ timing

Aucun enregistrement.

Framework openMairie Version 4.10.0-om5_no_code-1.2.0a1 | [Documentation](#) | [Forum](#) | [openMairie.org](#)

Il est possible de saisir et supprimer les triggers dans le formulaire des triggers :

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Pret Om5

Table Formulaire Action clé secondaire Procédures Triggers Vues

administration ➔ om_triggers

```
select proname FROM om5.om_proc WHERE SPLIT_PART(om_proc.proname, '_', 1)= 'pret'
```

Ajouter [Retour](#)

nom declencheur * pret_4

sur evenement * choisir un evenement ▼

sur la table pret

Si (condition)

executer la procédure * choisir une procédure ▼

sur enregistrement courant ▼

quand ? avant enregistrement ▼

Ajouter [Retour](#)

Note : Il faut créer la procédure avant le trigger qui lance la procédure.

Les options de l'assistant trigger sont les suivantes :

2.5.1 nom du trigger

Le nom du trigger est automatique : nom de la table + numéro d'ordre.

2.5.2 Événement :

Le trigger s'exécute lors de :

- la création d'enregistrement (insert)
- la modification d'un enregistrement (update)
- la destruction d'un enregistrement (delete)

2.5.3 table

Par défaut la table du formulaire

2.5. Saisir des triggers :

Non modifiable

2.5.4 ordre

ordre d'exécution des trigers : non implémenté

2.5.5 condition

condition d'exécution du trigger : non implémenté

2.5.6 Procédure

Choix de la procédure de la table (voir chapitre procédure)

2.5.7 sur ?

Une seule option : enregistrement courant (for each row).

L'option traitement n'est pas implémenté (for each statement)

2.5.8 quand ?

avant enregistrement : insert, update

après enregistrement : insert, update ou delete

L'option INSTEAD OF n'est pas implémentée.

2.6 Saisir les procédures :

Note : Attention ce formulaire est en cours de développement. Les procédures seront liés aux triggers postgresql de la table.

Il est possible de lister les procédures dans le menu om-no-code -> option om_tables - onglet procédures

Om5 No Code → Gestion De Tables Et Génération Des Objets → Pret Om5

Table Formulaire Action clé secondaire Procédures Triggers Vues

1 - 3 enregistrement(s) sur 3

+	nom	declare	begin
	pret_1	DECLARE nb_pret integer;	nb_pret:=(select count(*) from pret where emprunteur = NEW.emprunteur); update emprunteur set nombre_pret=nb_pret where emprunteur = NEW.emprunteur; RETURN NEW; END;
	pret_3	DECLARE nb TEXT;	update emprunteur set nombre_pret=coalesce(nombre_pret,0)-1 where emprunteur = OLD.emprunteur; RETURN OLD; END;
	pret_2	DECLARE nb TEXT;	update om5.emprunteur set nombre_pret=coalesce(nombre_pret,0)+1 where emprunteur = NEW.emprunteur; RETURN NEW; END;

Framework openMairie Version 4.10.0-om5_no_code-1.2.0a1 | [Documentation](#) | [Forum](#) | [openMairie.org](#)

Il est possible de saisir, modifier et supprimer les procédures dans le formulaire des procédures :

om5 no code → gestion de procédure stockée postgresql → pret_1

Modifier Retour

* pret_1

+ Déclaration de variable

Procedure

assistant

choisir un champ NEW ▾

choisir un champ OLD ▾

choisir un champ autre ▾

Procedure

```
nb_pret:=(select count(*) from pret where emprunteur = NEW.emprunteur);
update emprunteur set nombre_pret=nb_pret where emprunteur =
NEW.emprunteur;
```

Retour Retour mise à jour (NEW) ▾

{search_path=om5, public, pg_catalog}

Modifier Retour

2.6.1 nom du procedure

Le nom de la procédure est automatique : nom de la table + numéro d'ordre.

2.6.2 table

Par défaut la table du formulaire Non modifiable

2.6.3 Déclaration de variable

Champs optionnel utile si il est utilisé des variables internes :

2.6.4 Procédure :

Opération de la procédure

2.6.5 Retour

- retour mise à jour (NEW) pour INSERT et UPDATE. NEW is NULL dans un DELETE triggers
- retour delete (OLD) pour UPDATE ou DELETE. OLD is NULL dans un INSERT triggers

2.6.6 automaticité du code

Une partie du code est créé automatique et est complété par les champs : declare, procédure et retour.

exemple : calcul du nombre de prêt par emprunteur

```
-- automatique (nom de la procédure)
CREATE OR REPLACE FUNCTION om5.pret_1()
RETURNS trigger AS
$BODY
$DECLARE

-- champ déclare :
nb_pret integer;

-- automatique
BEGIN

-- champ procédure :
nb_pret:=(select count(*) from pret where emprunteur = NEW.emprunteur);
update emprunteur set nombre_pret=nb_pret where emprunteur = NEW.emprunteur;

-- champ retour (select):
RETURN NEW;

-- automatique : langage et schéma par défaut :
END;
$BODY$
LANGUAGE plpgsql VOLATILE
COST 100;
ALTER FUNCTION om5.pret_1() SET search_path=om5, public, pg_catalog;
```

2.6.7 Assistant :

L'assistant permet de pouvoir récupérer les champs de la table courante :

- NEW : valeur après la mise à jour
- OLD ; valeur avant la mise à jour

et les champs de la base.

2.7 Saisir des vues :

Note : Attention ce formulaire est en cours de développement. Il est prévu une aide à la construction de vue notamment d'agrégat pour les listes

Il est possible de lister les triggers dans le menu om-no-code -> option om_tables - onglet vues

table_name	table_schema	view_definition
emprunteur_1	om5	SELECT emprunteur.nom, emprunteur.prenom, count(pret.pret) AS pret FROM (om5.pret LEFT JOIN om5.emprunteur ON ((pret.emprunteur = emprunteur.emprunteur))) GROUP BY emprunteur.emprunteur HAVING (((emprunteur.prenom)::text = 'jean'::text) AND (emprunteur.age < 100)) ORDER BY emprunteur.nom;

Il est possible de saisir, modifier et supprimer les vues dans le formulaire des vues :

om5 no code > gestion des vues postgresql > emprunteur_1 om5

Modifier [Retour](#)

table_name * emprunteur_1
table_schema om5

select emprunteur.nom, emprunteur.prenom, count(pret.pret) AS pret

from om5.pret
LEFT JOIN om5.emprunteur ON pret.emprunteur = emprunteur.emprunteur

group_by emprunteur.emprunteur

having emprunteur.prenom = 'jean' AND emprunteur.age < 100

where

order_by emprunteur.nom

Modifier [Retour](#)

2.7.1 nom de la vue

Le nom de la vue est automatique : nom de la table + numéro d'ordre.

2.7.2 table

Par défaut la table du form

ulaire Non modifiable

2.7.3 assistant select

Liste des champs à sélectionner séparés par une virgule

```
livre.titre, livre.auteur, count(pret.pret) AS pret
```

Il est possible d'utiliser l'assistant pour rechercher les champs de la base :

2.7.4 assistant from

Tables de la vue, par défaut la table courante.

```
om5.pret
JOIN om5.livre ON pret.livre = livre.livre
join om5.evaluation ON livre.evaluation=evaluation.evaluation
```

Il est possible d'utiliser l'assistant pour construire les jointures :

La jointure de un à un est une jointure de la table en cours sur une autre table.

La jointure de un à plusieurs est une jointure d'une autre table sur la table en cours.

exemple :

- prêt est une jointure de un à plusieurs sur livre
- evaluation est une jointure de un à un sur livre

2.7.5 assistant group_by

Dans le cas d'une requête d'agrégation

```
livre.livre, livre.titre, livre.auteur
```

Il est possible d'utiliser l'assistant pour rechercher les champs de la base :

2.7.6 assistant having ou where

Having dans le cas d'une requête d'agrégation ou where dans les autres cas
les conditions sql sont séparées par les opérateurs AND, OR et/ou des parenthèses

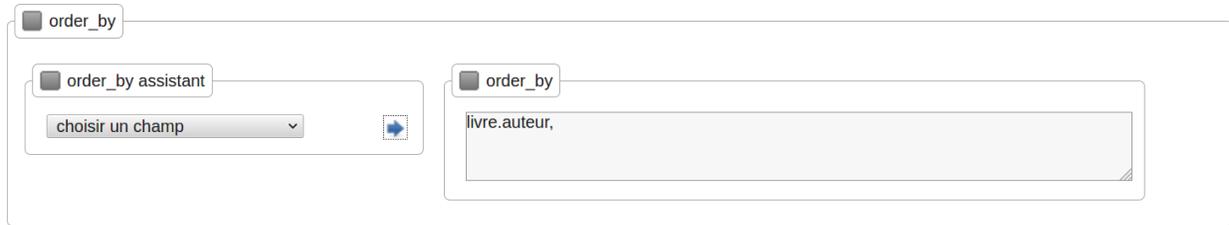
Il est possible d'utiliser l'assistant pour rechercher les champs de la base :

2.7.7 assistant order_by

Tri des enregistrements

Champs séparés par une virgule

Il est possible d'utiliser l'assistant pour rechercher les champs de la base :



Note : Si vous utilisez l'assistant, la dernière virgule et le dernier retour charriot sont enlevés par traitement pour les champs select, group_by et order_by

2.7.8 action « créer un widget » :

Il est possible de créer un widget dans le tableau de bord si celui-ci n'est pas existant

Le widget est créé dans administration -> widget :



Il est possible de modifier le titre. (l'argument indique la vue concernée à afficher)

Administration ➔ Tableaux De Bord ➔ Widget ➔ 9 Vue Emprunteur_1

widget tableau de bord

Modifier  Retour

widget * 9

libellé *

type *

script *

arguments

Modifier  Retour

Avec la composition du tableau de bord, le widget peut s'afficher dans le profil concerné :

Tableau De Bord

vue livre_1			vue emprunteur_1		
titre	auteur	pret	nom	prenom	pret
une vie	Maupassant	2	DUPONT	Jean	3
Pensées	Pascal	7	DURANT	Pierre	5
2 enregistrement(s)			GILBERT	Jean	1
			3 enregistrement(s)		

Framework openMairie Version 4.10.0-om5_no_code-1.2.0a1 | [Documentation](#) | [Forum](#) | [openMairie.org](#)

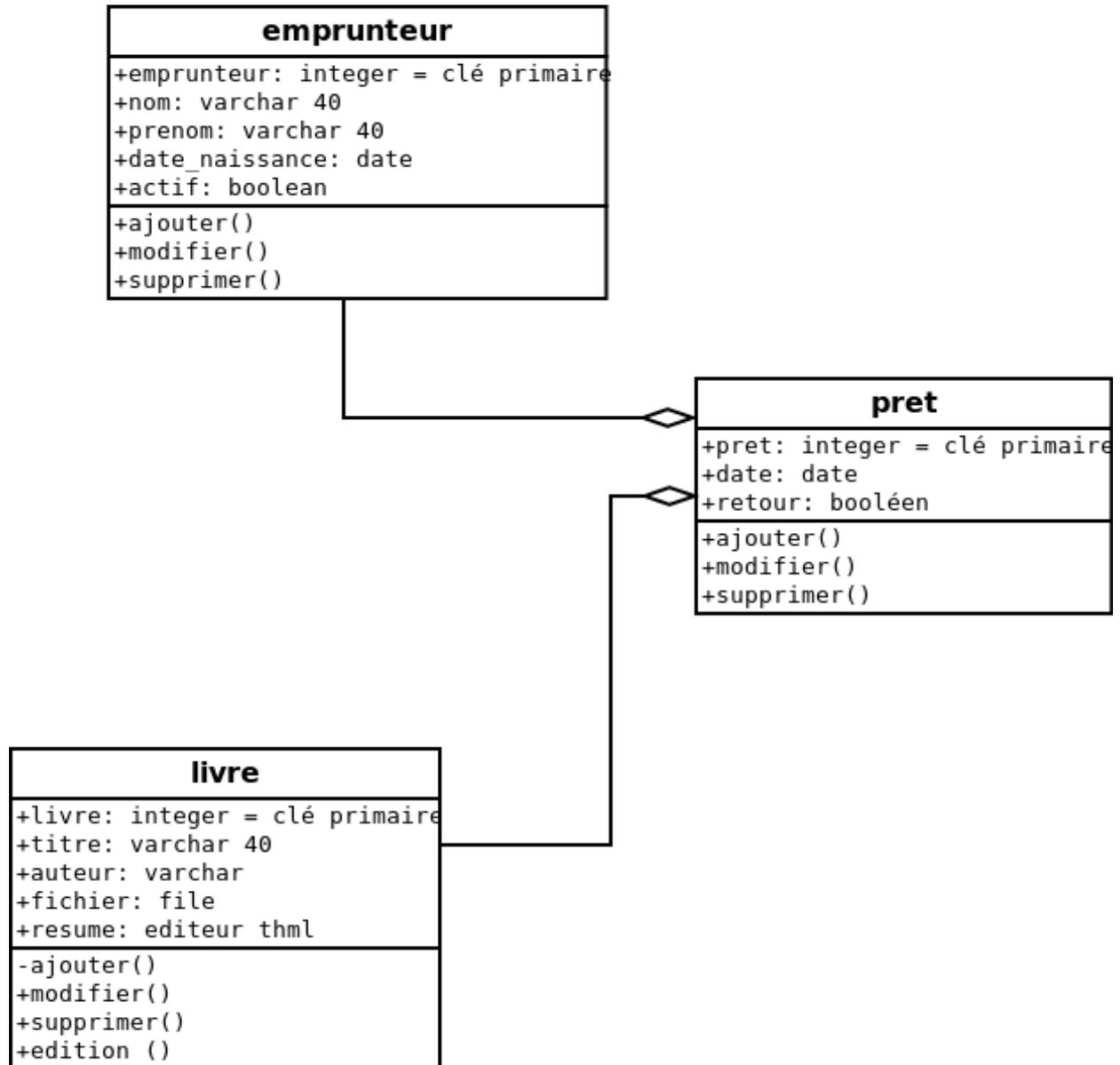
2.8 Exemple : la gestion de ma bibliothèque :

Note : Le sql de l'exemple est dans le fichier data/pgsql/om5_exemple.sql

Il est proposé à titre d'exemple, de créer l'application ma bibliothèque et le prêt à mes amis :

- dans un premier temps, on analyse ce que l'on veut faire et on peut produire un diagramme de classe
- on crée les tables avec om_tables en modifiant éventuellement les paramètres d'om_tables_parametre
- on crée les champs correspondants avec om_champs en modifiant éventuellement les paramètres d'om_forms
- on compose le formulaire avec l'option composeur du menu om5-no-code
- on génère les formulaires pour chaque table.
- on peut générer une action « générer un état » (sous formulaire action)
- on peut générer un sous état (sous formulaire champs)
- il est possible de créer des triggers et des procédures
- il est possible de créer des vues et créer des widgets dans le tableau de bord

2.8.1 Diagramme de classe de la gestion de bibliothèque :



2.8.2 La création des tables :

Les tables suivantes ont été créées avec `om_tables` : `livre`, `emprunteur`, `pret`

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets

Ce listing décrit les tables existantes pour votre application

Table								
1 - 3 enregistrement(s) sur 3				Tous		Recherche		
+	table	libelle	col1	col2	col3	menu	nb_col	recherche
	livre	Ma bibliothèque	Titre	Résumé	colonne 3		1	2
	emprunteur	Mes Amis	Etat Civil	colonne 2	colonne 3		1	1
	pret	prêt	livres ou amis	Retour	colonne 3		0	2

2.8.3 Les champs suivants ont été créés pour chaque table :

livre

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Livre Om5

Table													
Formulaire		Action		clé secondaire		Trigger pgsq							
1 - 4 enregistrement(s) sur 4													
+	champ	table	type	clé secondaire	obligatoire	libelle	type	bloc	no	calcul	liste	no_l	lib_l
	livre.titre	livre	character varying		Oui	titre	C1		1		1	1	titre
	livre.auteur	livre	character varying		Oui	auteur	C1		2		1	2	auteur
	livre.fichier	livre	character varying		Non	fichier	file	C1	3		1	3	fichier
	livre.resume	livre	text		Non		html	C2	1		1	4	resumé

emprunteur

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Emprunteur Om5

Table													
Formulaire		Action		clé secondaire		Trigger pgsq							
1 - 4 enregistrement(s) sur 4													
+	champ	table	type	clé secondaire	obligatoire	libelle	type	bloc	no	calcul	liste	no_l	lib_l
	emprunteur.actif	emprunteur	boolean		Non	actif		C2	1		1	4	actif
	emprunteur.date_de_naissance	emprunteur	date		Non	date de naissance		C1	3		1	3	date de naissance
	emprunteur.nom	emprunteur	character varying		Oui	nom		C1	1		1	1	nom
	emprunteur.prenom	emprunteur	character varying		Oui	prénom		C1	2		1	2	prénom

pret

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Pret Om5

Table Formulaire Action clé secondaire Trigger pgsq

1 - 4 enregistrement(s) sur 4

+	▶ champ	▶ table	▶ type	▶ clé secondaire	▶ obligatoire	▶ libelle	▶ type	▶ bloc	▶ calcul	▶ liste	▶ no_l	▶ lib_l
	pret.emprunteur	pret	integer	pret_emprunteur_fkey	Oui	emprunteur	C1	1		1	1	emprunteur
	pret.livre	pret	integer	pret_livre_fkey	Oui	livre	C1	2		1	2	livre
	pret.date_de_pret	pret	date		Oui	date de prêt	C1	3		1	3	date de prêt
	pret.retour	pret	boolean		Non	retour	C1	4		1	4	retour

Les contraintes de clé secondaires sont visibles dans le sous formulaire om_contrainte de om_tables (option du menu pg_admin) :

- pret_livre_fkey
- pret_emprunteur_fkey

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Pret Om5

Table Formulaire Action clé secondaire Trigger pgsq

1 - 2 enregistrement(s) sur 2

	▶ contrainte	▶ schema	▶ table	▶ champ	▶ foreign table	▶ foreign champ
	pret_emprunteur_fkey	om5	pret	emprunteur	emprunteur	emprunteur
	pret_livre_fkey	om5	pret	livre	livre	livre

Note : Les sous formulaires sont créés par les clés secondaires.

2.8.4 Les formulaires ont été composés de la manière suivante :

Pour la table emprunteur :

Om5 No Code ➔ Formulaire

Composition du formulaire

Formulaire pour la table

emprunteur.nom — ✕	emprunteur.aktif — ✕
character varying -	boolean -
emprunteur.prenom — ✕	
character varying -	
emprunteur.date_de_naissance — ✕	
date -	

pour la table livre :

Om5 No Code ➔ Formulaire

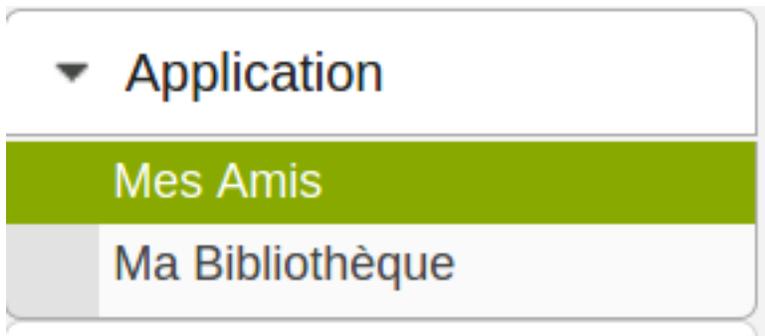
Composition du formulaire

Formulaire pour la table

livre.titre - x character varying -	livre.resume - x text - html
livre.auteur - x character varying -	
livre.fichier - x character varying - file	

2.8.5 Le menu application :

Le menu option application est créé au fur et à mesure de la création des tables :



2.8.6 Les formulaires et le sous formulaire généré :

Il faut ensuite générer le formulaire avec chaque table (voir action générer d'om_table).

Il est possible de lister les emprunteurs dans le menu application -> option « mes amis »

Application ➔ Emprunteur

emprunteur

Afficher la recherche simple

nom prénom date de naissance de à actif

Utilisation de * pour zones de saisie: A*D peut correspondre à 'ABCD'. Par défaut * est ajouté au debut et à la fin des recherches.

Recherche

1 - 2 enregistrement(s) sur 2

	▶ emprunteur ⌵	▶ nom ⌵	▶ prénom ⌵	▶ date de naissance	▶ actif
	2	DUPONT	Jean	02/05/2003	Oui
	1	DURANT	Pierre	01/05/2007	Oui

Il est possible de saisir, modifier et supprimer les emprunteurs dans le formulaire suivant :

Application ➔ Emprunteur ➔ 2 DUPONT

emprunteur

Retour

2

Etat Civil

nom DUPONT

prénom Jean

date de naissance 02/05/2003

Actif
actif Oui

modifier
 supprimer

Retour

Il est possible de lister les livres dans le menu application -> option « ma bibliothèque »

Application ➔ Livre

livre

1 - 2 enregistrement(s) sur 2 Recherche

	▶ livre ⌵	▶ titre ⌵	▶ auteur ⌵	▶ fichier ⌵
	1	Les pensées	PASCAL	
	2	Une vie	Maupassant	

Framework openMairie Version 4.10.0-om5_RAD-1.1.0 | [Documentation d'om5_RAD](#) | [Forum](#) | [openMairie.org](#)

Il est possible de saisir, modifier et supprimer les livres dans le formulaire suivant :

Application ➔ Livre ➔ 1 Pensées

livre pret

[Retour](#)

1

Titre
titre Pensées
auteur Pascal
fichier json_type_om.txt [Visualiser](#) ➔ [Télécharger](#)

[modifier](#)
[supprimer](#)
[Livres en prêt](#)

Résumé
Les **Pensées de Pascal** sont un des rares textes du XVIIe siècle dont nous possédions les manuscrits originaux. Cela tient au fait que l'œuvre est demeurée inachevée en raison de la mort prématurée de son auteur. Dans les années qui ont suivi la polémique des *Provinciales* et le concours sur la roulette,

[Retour](#)

Il est possible de lister les livres empruntés dans le menu application -> dans le sous formulaire de livre ou d'emprunteur :

Application ➔ Livre ➔ 1 Pensées

livre pret

1 - 1 enregistrement(s) sur 1

	▶ pret	▶ emprunteur	▶ livre	▶ date de prêt	▶ retour
	1	DUPONT	Pensées	09/05/2024	Non

Il est possible de saisir, modifier et supprimer les emprunts de livre dans le formulaire généré :

Application ➔ Livre ➔ 1 Pensées

livre pret

application ➔ pret

[Ajouter](#) [Retour](#)

livres ou amis

emprunteur * DUPONT

livre * Pensées

date de prêt * 09/05/2024

Retour

[Ajouter](#) [Retour](#)

2.8.7 La génération des éditions :

Voir le chapitre précédent : « saisie des actions »

Application ➔ Livre ➔ 1 Pensées

livre pret

Retour

1

Titre
titre Pensées
auteur Pascal
fichier json_type_om.txt Visualiser ➔ Télécharger

modifier
supprimer
Livres en prêt

Résumé
Les **Pensées de Pascal** sont un des rares textes du XVIIe siècle dont nous possédions les manuscrits originaux. Cela tient au fait que l'œuvre est demeurée inachevée en raison de la mort prématurée de son auteur. Dans les années qui ont suivi la polémique des *Provinciales* et le concours sur la roulette,

Retour

Il est possible en cliquant sur l'action « livre en prêt » d'accéder à l'état suivant :



le 14/05/2024

Titre :

Pensées

Auteur :

Pascal

Résumé :

Les **Pensées de Pascal** sont un des rares textes du XVIIe siècle dont nous possédions les manuscrits originaux. Cela tient au fait que l'œuvre est demeurée inachevée en raison de la mort prématurée de son auteur. Dans les années qui ont suivi la polémique des *Provinciales* et le concours sur la roulette,

2.8.8 Les procédures

Exemple de procédures créées avec les formulaires om5

Calcul d'âge

```

CREATE OR REPLACE FUNCTION om5.emprunteur_1()
  RETURNS trigger AS
$BODY
$DECLARE
BEGIN
NEW.age = date_part('year', age(NEW.date_de_naissance));
RETURN NEW;
END;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION om5.emprunteur_1() SET search_path=om5, public, pg_catalog;

```

Nombre de prêt par emprunteur

```

CREATE OR REPLACE FUNCTION om5.pret_1()
  RETURNS trigger AS
$BODY
$DECLARE
nb_pret integer;
BEGIN
nb_pret:=(select count(*) from pret where emprunteur = NEW.emprunteur);
update emprunteur set nombre_pret=nb_pret where emprunteur = NEW.emprunteur;
RETURN NEW;
END;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION om5.pret_1() SET search_path=om5, public, pg_catalog;

*** OU :

CREATE OR REPLACE FUNCTION om5.pret_2()
  RETURNS trigger AS
$BODY
$DECLARE
nb TEXT;
BEGIN
update om5.emprunteur set nombre_pret=coalesce(nombre_pret,0)+1 where emprunteur =
↳NEW.emprunteur;
RETURN NEW;
END;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION om5.pret_2() SET search_path=om5, public, pg_catalog;

*** ET (en delete)

CREATE OR REPLACE FUNCTION om5.pret_3()
  RETURNS trigger AS
$BODY$DECLARE
nb TEXT;
BEGIN
update emprunteur set nombre_pret=coalesce(nombre_pret,0)-1 where emprunteur = OLD.
↳emprunteur;
RETURN OLD;

```

(suite sur la page suivante)

```
END; $BODY$
LANGUAGE plpgsql VOLATILE
COST 100;
ALTER FUNCTION om5.pret_3() SET search_path=om5, public, pg_catalog;
```

2.8.9 Triggers pour calcul d'âge

Exemple de triggers créés avec les formulaires om5

```
*** en insert

CREATE TRIGGER emprunteur_1
  BEFORE INSERT
  ON om5.emprunteur
  FOR EACH ROW
  EXECUTE PROCEDURE om5.emprunteur_1();

*** en update

CREATE TRIGGER emprunteur_2
  BEFORE UPDATE
  ON om5.emprunteur
  FOR EACH ROW
  EXECUTE PROCEDURE om5.emprunteur_1();
```

2.8.10 Vues pour tableau de bord : nombre de prêt

Exemple de vues créés avec les formulaires om5

```
*** par emprunteur

CREATE OR REPLACE VIEW om5.emprunteur_1 AS
  SELECT emprunteur.nom,
         emprunteur.prenom,
         count(pret.pret) AS pret
  FROM om5.pret
        LEFT JOIN om5.emprunteur ON pret.emprunteur = emprunteur.emprunteur
  GROUP BY emprunteur.emprunteur
  HAVING emprunteur.age < 25
  ORDER BY emprunteur.nom;

*** par livre

CREATE OR REPLACE VIEW om5.livre_1 AS
  SELECT livre.titre,
         livre.auteur,
         count(pret.pret) AS pret
  FROM om5.pret
        JOIN om5.livre ON pret.livre = livre.livre
  GROUP BY livre.livre, livre.titre, livre.auteur;
```

Il est présenter dans ce chapitre la description technique du projet dans le cadre du framework openMairie :

3.1 Le framework openMairie :

Le projet om5-no-code est un projet de développement rapide qui permet une prise en main rapide du framework.

Il est composé de 2 modules :

- un installateur du framework,
- les formulaires de l'option om5-no-code qui permet de créer les tables et de générer des formulaires et des éditions.

Le but est de créer rapidement une mini application ou une maquette sans écrire une seule ligne de code.

Le public visé est les développeurs peu expérimentés ou débutant, en proposant par la suite une montée en puissance pour maîtriser l'ensemble du framework (formation, assistance technique, forum ...). Il s'agit de mettre en place un développement « no code » accessible avec une connaissance de modèle de données mais aussi un modèle de développement pour les utilisateurs avancés car om5-no-code veut être aussi une alternative aux :

- applications sur tableur qui deviennent souvent complexes avec beaucoup de colonnes et de nombreuses redondances
- aux gestionnaires de contenu qui n'intègrent pas les règles du modèle de CODD des bases de données (cle/valeur, redondances)
- aux systèmes d'information géographique qui n'intègrent pas la dimension relationnelle

Ce projet qui prolonge le module du générateur, veut s'intégrer dans les développements futurs d'openMairie :

- poc_de_query : pour le passage aux versions php supérieures à 8.0 et faciliter l'installation des composants
- poc_view_js : pour l'utilisation de librairie d'affichage java script plus moderne.

3.1.1 Le projet ne modifie pas le core :

Le projet utilise le framework version 4.10.0 et ne modifie aucune classe du core.

Le projet modifie seulement 4 fichiers :

- le fichier `index.php` à la racine car il appelle l'installateur dans le répertoire `app` (`om_setup_config.php`), si le fichier `dyn/database.inc.php` n'existe pas,
- le fichier `data/pgsql/install.sql` avec l'ajout de la création des vues : `init_gen.sql`,
- le fichier `framework_openmairie.class.php` par surcharge de la méthode `set_config__menu()` en ajoutant 2 options :
 - l'option « application » qui présente la liste des tables générées par l'utilisateur,
 - l'option « om5-no-code » qui appelle les formulaires d'administration de la base de données : `om_tables`, avec les formulaire `om_champs` et `om_contrainte`

Les autres ajouts sont uniquement des objets générés et leurs surcharges.

3.1.2 Le projet se base sur des vues sur `information_schema` :

Il est créé dans le répertoire `data/pgsql` 3 vues sur `information_schema` :

- `om_tables` : cette vue contient les tables du schéma sauf celles préfixées par `om`
- `om_champs` : cette vue contient les champs des tables ci dessus
- `om_contraintes` : cette vue contient les clés secondaires des tables ci dessus
- `om_vues` : cette vue contient les vues sur les tables ci dessus
- `om_triggers` : cette vue contient les triggers des tables ci dessus
- `om_proc` : cette vue contient les procédures utilisées par les tables ci dessus

Les répertoires de `gen` sont rajoutés avec les classes relatives aux vues qui ont été créés par le générateur :

- dans `gen/obj` : `om_tables.class.php`, `om_champs.class.php`, `om_contraintes.class.php`, `om_triggers.class.php`, `om_proc.class.php`, `om_vues.class.php`
- dans `gen/sql/pgsql` : `om_tables.inc.php`, `om_champs.inc.php`, `om_contraintes.inc.php`, `om_triggers.inc.php`, `om_proc.inc.php`, `om_vues.inc.php`

Enfin ces classes sont surchargées dans répertoires `obj` et `sql/pgsql`.

Le principe est simple : les classes sont liées aux vues d'`information_schema` dans l'affichage.

La mise à jour d'`information_schema` se fait avec :

- les commandes de création et suppression de table pour `om_table`,
- les commandes de création, modification et suppression de champs pour `om_forms`
- la commande de suppression de clé secondaire pour `om_contraintes`.
- les commandes de création, modification et suppression de vues pour `om_vues`
- les commandes de création, modification et suppression de trigger pour `om_triggers`
- les commandes de création, modification et suppression de procédures pour `om_proc`

Note : Les paramètres particuliers des tables pour `gen_plus.class.php` sont stockés dans la table `om_tables_parametres`. Les paramètres particuliers des champs pour `gen_plus.class.php` sont stockés dans la table `om_forms` Les paramètres particuliers des actions pour `gen_plus.class.php` sont stockés dans la table `om_actions`

3.2 La surcharge `om_gen_plus.class.php` :

Il a été surchargé `om_gen.class.php` par `om_gen_plus.class.php` dans le repertoire `app`.

`om_gen_plus` prend en compte le paramétrage de `om_tables_parametre` et de `om_forms`.

Les méthodes suivantes ont été surchargées :

- `def_obj_meth_setlib` : pour le paramétrage des libellés de champs
- `def_obj_meth_settype_by_maj` : pour le paramétrage des types de champs
- `def_obj_meth_get_var_sql_forminc` pour le paramétrage de l'ordre des champs
- `def_php_script_header` : pour la mise a jour du header avec `genplus`
- `stream_slightly_equals_file` : la génération se fait à chaque fois et non quand la base est changée

— table_obj_class_gen : pour ajouter de la méthode setLayout

Il a été ajouter les méthodes suivantes :

- def_obj_meth_setlayout : définition de la méthode setlayout pour regrouper les champs
- def_moteur_recherche_inc : définition du moteur de recherche
- def_obj_meth_init_class_actions() : définition des actions
- edition_action() : méthode d'édition.

CHAPITRE 4

Bibliographie

— <http://www.openmairie.org/>

CHAPITRE 5

Contributeurs

(par ordre alphabétique)
— François Raynaud