
openMairie Framework Documentation

Version 4.5

openMairie

26 10 2019

Table des matières

1	Prérequis et installation	3
1.1	Pré-requis	3
1.2	Installation	4
1.2.1	Installation des fichiers du framework	4
1.2.1.1	Télécharger l'archive zip	4
1.2.1.2	Décompresser l'archive zip dans le répertoire de votre serveur web	4
1.2.2	Création et initialisation de la base de données	4
1.2.2.1	Créer la base de données	4
1.2.2.2	Initialiser la base de données	4
1.2.3	Configuration de l'applicatif	5
1.2.3.1	Création des répertoires dédiés au stockage des données de l'applicatif	5
1.2.3.2	Positionner les permissions nécessaires au serveur web	5
1.2.3.3	Configuration de la connexion à la base de données	5
1.3	Connexion au framework	5
1.3.1	Ouverture dans le navigateur	5
1.3.2	Login	5
1.4	En cas d'erreur	6
1.4.1	Activer le mode debug	6
2	Tutoriel - Créer une application	7
2.1	Créer la base de données	7
2.2	Créer les formulaires	9
2.2.1	Générer les formulaires et édition du courrier	9
2.2.2	Générer les formulaires et édition de l'émetteur	12
2.2.3	Générer les formulaires et édition de service	12
2.2.4	Intégrer les formulaires dans le menu	13
2.2.5	Menu	14
2.3	Personnaliser son application	17
2.3.1	Faire un affichage courrier plus convivial	18
2.3.2	Rendre obligatoire des champs	19
2.3.3	Valoriser un champ par défaut	20
2.3.4	Mettre en majuscule un champ	20
2.3.5	Principe à retenir	21
2.4	Modifier la base et régénérer	21
2.4.1	Rajouter un champ registre dans courrier	21
2.4.2	Rajouter l'adresse dans émetteur	22

2.4.3	Améliorer la présentation du formulaire emetteur	22
2.4.4	Les surcharges d'openCourrier	23
2.5	Créer ses états	23
2.5.1	Créer l'état service	23
2.5.2	Créer le sous-état courrier	25
2.5.3	Associer le sous-état <i>courrier</i> à l'état <i>service</i>	25
2.5.4	Mettre le nom et le prénom de l'emetteur dans le sous-état	27
3	Manuel d'usage	31
3.1	Ergonomie	31
3.1.1	Ergonomie générale	31
3.1.1.1	Le logo	31
3.1.1.2	Les actions personnelles	31
3.1.1.3	Les raccourcis	33
3.1.1.4	Le menu	33
3.1.1.5	Les actions globales	33
3.1.2	Usage du module om_sig	34
3.1.2.1	Ergonomie de l'interface SIG interne :	34
3.1.2.2	les fonds :	35
3.1.2.3	les couches :	35
3.1.2.4	Information :	36
3.1.2.4.1	un fond wms	36
3.1.2.4.2	le marqueur	37
3.1.2.4.3	une donnée vecteur	38
3.1.2.5	Boite à outils :	38
3.1.2.5.1	Accès au formulaire de saisie de données	39
3.1.2.5.2	Navigation	40
3.1.2.5.3	Se géolocaliser dans la carte	40
3.1.2.5.4	Mesurer une distance	40
3.1.2.5.5	Mesurer une aire	41
3.1.2.6	Mode édition :	41
3.1.2.7	Edition d'un point :	41
3.1.2.7.1	Création d'un ou plusieurs points :	41
3.1.2.7.2	Modifier une géométrie sélectionnées :	42
3.1.2.7.3	(Dé)selectionner une géométrie :	42
3.1.2.7.4	Supprimer une géométrie selectionner :	42
3.1.2.7.5	Vérifier avant enregistrement d'un point :	42
3.1.2.7.6	Enregistrer un point :	43
3.1.2.8	Edition d'un polygone :	43
3.1.2.8.1	Utilisation du panier pour construire une géométrie	43
3.1.2.8.2	Créer un polygone	45
3.1.2.8.3	Modifier un polygone sélectionné :	45
3.1.2.8.4	(Dé)selectionner une géométrie :	45
3.1.2.8.5	Supprimer un polygone selectionné :	46
3.1.2.8.6	Vérifier avant enregistrement d'un polygone :	46
3.1.2.8.7	Enregistrer un polygone :	46
3.1.2.9	Modification d'un ligne :	46
3.1.2.9.1	Utilisation du panier pour construire une géométrie	46
3.1.2.9.2	Créer une ligne :	46
3.1.2.9.3	Modifier une ligne sélectionnée :	47
3.1.2.9.4	(Dé)selectionner une géométrie :	47
3.1.2.9.5	Supprimer une ligne selectionnée :	47
3.1.2.9.6	Vérifier avant enregistrement d'une ligne :	47
3.1.2.9.7	Enregistrer une ligne :	47

3.2	Administration	47
3.2.1	Les tableaux de bord	48
3.2.1.1	Widget	48
3.2.1.2	Composition	48
3.2.2	Les éditions	48
3.2.2.1	États et lettres types	48
3.2.2.1.1	Positionnement des éléments dans l'édition	48
3.2.2.1.2	Bloc "édition"	49
3.2.2.1.2.1	Paramètres généraux de l'édition	49
3.2.2.1.3	Bloc "en-tête"	50
3.2.2.1.4	Bloc "titre"	51
3.2.2.1.4.1	Paramètres du titre de l'édition	51
3.2.2.1.5	Bloc "corps"	52
3.2.2.1.5.1	Paramètres des sous-états	52
3.2.2.1.6	Bloc "pied de page"	52
3.2.2.1.7	Bloc "champs de fusion"	53
3.2.2.1.8	Aide à la saisie dans les éditeurs de texte riche	53
3.2.2.1.8.1	Configurations disponibles	53
3.2.2.1.8.2	Gestion des tableaux	53
3.2.2.1.8.3	Tableaux insécables	55
3.2.2.1.8.4	Gestion des sauts de page	55
3.2.2.1.8.5	Gestion des code-barres	56
3.2.2.1.8.6	Majuscule/Minuscule	56
3.2.2.1.8.7	Gestion du mode plein écran	56
3.2.2.1.8.8	Code source	56
3.2.2.1.8.9	Correction orthographique	56
3.2.2.1.8.10	Insertion de sous-états	56
3.2.2.2	Sous-états	57
3.2.2.3	Requêtes	57
3.2.2.4	Logos	57
3.2.3	Paramétrage du module om_sig	57
3.2.3.1	Principes	57
3.2.3.1.1	Pré requis	57
3.2.3.1.2	Géo localisation automatique sur une adresse postale	57
3.2.3.1.3	Affichage de carte	58
3.2.3.1.4	L'intégration dans openMairie	58
3.2.3.1.5	Paramétrage de la carte	58
3.2.3.1.6	Autre point d'entrée	59
3.2.3.2	Saisie des cartes :	63
3.2.3.2.1	Formulaire	63
3.2.3.2.2	Description des champs :	64
3.2.3.3	Saisie des flux :	65
3.2.3.3.1	Formulaire	65
3.2.3.3.2	Description des champs :	66
3.2.3.4	Saisie des géométries :	67
3.2.3.4.1	Formulaire	67
3.2.3.4.2	Description des champs :	68
3.2.3.5	Saisie des flux de la carte :	68
3.2.3.5.1	Formulaire :	69
3.2.3.5.2	Description des champs :	69
3.2.3.6	Saisie om_sig_extent	70
3.2.3.6.1	Formulaire	71
3.2.3.6.2	Les extent par défaut	71
3.2.3.6.3	Construction d'un extent avec openStreetMap :	71

3.3	Génération	72
3.3.1	Introduction	72
3.3.2	L'interface	73
3.3.2.1	Analyse de la base	74
3.3.2.2	Les fichiers à générer	74
3.3.3	Conditions de génération	75
3.3.3.1	Contraintes de la base de données	75
3.3.3.2	Contraintes du système de fichiers	75
3.3.4	Définition des modèles de données	75
3.3.4.1	L'identifiant	75
3.3.4.1.1	Définition de l'identifiant	75
3.3.4.1.2	Fonctionnement interne du générateur	75
3.3.4.2	Les références vers d'autres objets	76
3.3.4.2.1	Définition des références	76
3.3.4.2.1.1	Avec PostgreSQL	76
3.3.4.2.2	Fonctionnement interne du générateur	76
3.3.4.2.3	Affichage dans les formulaires	76
3.3.4.3	Les champs uniques	76
3.3.4.3.1	Définition des champs uniques	77
3.3.4.3.2	Fonctionnement interne du générateur	77
3.3.4.3.3	Affichage dans les formulaires	77
3.3.4.4	Les champs requis	77
3.3.4.4.1	Définition des champs requis	77
3.3.4.4.2	Fonctionnement interne du générateur	77
3.3.4.4.3	Affichage dans les formulaires	77
3.3.4.5	Le champ libellé	78
3.3.4.5.1	Définition du libellé	78
3.3.5	Fonctionnalités avancées	78
3.3.5.1	Ajouter une date de validité à un modèle	78
3.3.5.1.1	Description	78
3.3.5.1.2	Définition des dates de validité	78
3.3.5.1.3	Affichage dans les formulaires	79
3.3.6	L'analyse de la base	79
3.3.6.1	Type de champs	79
3.3.6.2	Equivalence type pgsql / type openMairie	80
3.3.6.3	Nom de champ et nom de table	80
3.3.7	Les fichiers générés	80
3.3.7.1	Formulaires	81
3.3.7.1.1	Paramètres de type table	81
3.3.7.1.2	Paramètres de type Form	81
3.3.7.2	Objets « métier »	81
3.3.7.3	Etats	83
3.3.7.4	Requêtes mémorisées	83
3.3.7.5	Imports	83
3.3.7.6	Mots-clefs Robot Framework	83
3.3.8	Paramétrage générateur	84
3.3.8.1	gen/dyn/gen.inc.php	84
3.3.8.2	gen/dyn/tab.inc.php	85
3.3.8.3	gen/dyn/form.inc.php	85
3.3.8.4	gen/dyn/permissions.inc.php	86
3.3.8.5	gen/dyn/pdf.inc.php	87
3.3.8.6	gen/dyn/etat.inc.php	88
3.3.8.7	gen/dyn/sousetat.inc.php	88
3.3.8.8	Limites du générateur	88

3.4	Intégration	89
3.4.1	Installation de qgis serveur sur linux debian et ubuntu	89
3.5	Pour aller plus loin.	89
3.5.1	Les scripts spécifiques de l'application	90
3.5.1.1	Introduction	90
3.5.1.2	Réaliser un script complémentaire	90
3.5.1.3	Exemple	92
4	Manuel de référence	95
4.1	Arborescence	96
4.1.1	Introduction	96
4.1.2	Les répertoires spécifiques à l'applicatif	97
4.1.2.1	data/	97
4.1.2.2	app/	97
4.1.2.3	dyn/	97
4.1.2.4	gen/	97
4.1.2.5	locales/	98
4.1.2.6	obj/	98
4.1.2.7	sql/	98
4.1.2.8	tests/	98
4.1.2.9	var/	98
4.1.3	Les répertoires du framework	99
4.1.3.1	php/	99
4.1.3.2	core/	99
4.1.3.3	css/	99
4.1.3.4	img/	99
4.1.3.5	js/	99
4.1.3.6	lib/	99
4.1.3.7	pdf/	100
4.1.3.8	scr/	100
4.1.3.9	spg/	100
4.2	Initialisation de la base de données	100
4.2.1	Description du dossier data/pgsql/	100
4.2.1.1	Description de tous les fichiers init*.sql	101
4.2.1.1.1	Le fichier init.sql	101
4.2.1.1.2	Les fichiers init_metier*.sql	101
4.2.1.1.3	Les fichiers init_parametrage*.sql	101
4.2.1.1.4	Le fichier init_data.sql	102
4.2.1.2	Description des fichiers vX.X.X.sql ou ver_X.X.X.sql	102
4.2.1.3	Description du fichier update_sequences.sql	102
4.2.1.4	Description du fichier install.sql	102
4.2.2	Paramétrage de la connexion à la base de données	102
4.3	Paramétrage du framework	103
4.3.1	Introduction	104
4.3.2	Les scripts de paramétrage	104
4.3.3	Le serveur d'envoi de mail	105
4.3.4	L'annuaire LDAP	105
4.3.5	Les zones de navigation	106
4.3.5.1	Le menu	107
4.3.5.2	Les actions personnelles	109
4.3.5.3	Les raccourcis	110
4.3.5.4	Les actions globales	111
4.3.6	Les variables locales et la langue	112
4.3.7	Le paramétrage de l'application metier	112

4.3.7.1	Le nom de l'application	113
4.3.7.2	Le titre HTML de l'application	114
4.3.7.3	Le nom de la session	115
4.3.7.4	Le mode démonstration	116
4.3.7.5	La redéfinition du mot de passe oublié par l'utilisateur	116
4.3.7.6	Le nombre de colonnes du tableau de bord	116
4.3.7.7	Le favicon de l'application	117
4.3.7.8	Le mode de gestion des permissions	118
4.3.7.9	La valeur par défaut lorsqu'une permission n'existe pas	119
4.3.7.10	Les extensions de fichiers autorisées	120
4.3.7.11	La taille maximale de fichiers autorisée	120
4.3.8	Le Parametrage des librairies	120
4.3.9	Le mode DEBUG	121
4.3.10	La version de votre application	121
4.3.11	Les informations generales	121
4.3.12	L'installation automatique	122
4.3.13	Les paramètres des combos	122
4.3.14	Les paramètres éditions	122
4.3.15	Les paramètres om_sig	122
4.4	La gestion des accès	123
4.4.1	Introduction	123
4.4.2	Fonctionnalités	123
4.4.3	Les tables	123
4.4.4	Les règles	124
4.4.5	La multi-collectivité	125
4.4.6	Les login et logout	125
4.4.7	Les utilitaires	125
4.5	Le tableau de bord	125
4.5.1	Principe	126
4.5.1.1	paramétrage	126
4.5.1.2	les widgets	126
4.5.1.3	le tableau de bord paramétrable	126
4.5.2	widget	126
4.5.2.1	la création de widget	126
4.5.2.2	Le widget de type "Web"	127
4.5.2.2.1	interne	127
4.5.2.2.2	externe	128
4.5.2.3	Le widget de type "Script"	129
4.5.2.4	Modèle de données	129
4.5.3	Les tableaux de bord	130
4.5.3.1	accès au tableau de bord	130
4.5.3.2	Modèle de données	132
4.5.4	Composants	132
4.6	Les listings	132
4.6.1	Introduction	133
4.6.2	Les scripts <i>scr/tab.php</i> et <i>scr/soustab.php</i>	134
4.6.3	Configuration	134
4.6.3.1	\$ent	135
4.6.3.2	\$serie	135
4.6.3.3	\$table	135
4.6.3.4	\$champAffiche	135
4.6.3.5	\$champRecherche	135
4.6.3.6	\$tri	136
4.6.3.7	\$selection	136

4.6.3.8	\$edition	136
4.6.3.9	\$tab_title	136
4.6.3.10	\$tab_description	136
4.6.3.11	\$tab_actions	136
4.6.4	Actions des tableaux	137
4.6.4.1	Les actions par défaut	138
4.6.4.2	Créer de nouvelles actions	138
4.6.4.2.1	Définition de l'action	138
4.6.4.2.2	Définition du mode d'affichage en sous-tableau	139
4.6.4.2.3	Définition de l'ordre d'affichage	139
4.6.4.2.4	Définition des droits d'affichage	139
4.6.5	Les fonctionnalités	139
4.6.5.1	Le tri	140
4.6.5.1.1	Premier clic pour le tri croissant	140
4.6.5.1.2	Second clic pour le tri décroissant	140
4.6.5.1.3	Troisième clic pour aucun tri particulier	141
4.6.5.2	L'export CSV	141
4.6.5.3	La recherche avancée	142
4.6.5.3.1	Les différents types de recherche	142
4.6.5.3.1.1	Recherche simple	142
4.6.5.3.1.2	Recherche avancée	142
4.6.5.3.1.3	Recherche avancée mono-critère	142
4.6.5.3.1.4	Recherche avancée multi-critères	142
4.6.5.3.2	Configuration de la recherche avancée	143
4.6.5.3.2.1	Activation	143
4.6.5.3.2.2	Autres paramètres	143
4.6.5.3.3	Configuration des critères de recherche	144
4.6.5.3.3.1	Configuration simple	144
4.6.5.3.3.2	Configuration avancée	146
4.6.5.3.3.3	Créer un intervalle de date	146
4.6.5.3.3.4	Créer un champ de recherche avec menu déroulant personnalisé	146
4.6.5.3.3.5	Tester si une donnée est présente ou non dans un groupe de données	147
4.6.6	Le composant openMairie	148
4.7	Les formulaires	148
4.7.1	Introduction	149
4.7.1.1	Consultation	149
4.7.1.2	Ajout	149
4.7.1.3	Modification	149
4.7.1.4	Suppression	150
4.7.1.5	Accès	150
4.7.2	Les scripts <i>scr/form.php</i> et <i>scr/sousform.php</i>	150
4.7.3	Configuration via le script <i>sql/pgsql/<OBJ>.inc.php</i>	151
4.7.3.1	\$ent	151
4.7.3.2	\$sousformulaire	151
4.7.3.3	\$sousformulaire_parameters	151
4.7.4	Configuration via le script <i>sql/pgsql/<OBJ>.form.inc.php</i>	152
4.7.4.1	\$form_title	152
4.7.5	Les fonctions	152
4.7.5.1	La fonction verrou	152
4.7.5.2	La fonction directlink	152
4.7.6	Actions du menu contextuel de la consultation	153
4.7.6.1	Définition des actions dans les attributs de la classe de l'objet	153
4.7.6.2	CRUD	154
4.7.6.3	Définition des actions dans <i>*.form.inc.php</i> (obsolète)	154

4.7.7	Description de la classe dbform	155
4.7.7.1	Présentation des méthodes de la classe	155
4.7.7.2	Méthodes d'initialisation de l'affichage du formulaire	155
4.7.7.3	Méthodes d'actions	158
4.7.7.4	Gestion des transactions lors de l'appel aux méthodes d'actions	159
4.7.7.5	Méthodes appelées lors de la validation	159
4.7.7.6	Méthodes permettant d'afficher des informations spécifiques.	159
4.7.8	Description de la classe formulaire	160
4.7.8.1	Méthodes d'affichage de widgets	160
4.7.8.2	Les méthodes de construction et d'affichage	162
4.7.8.3	Les méthodes assesseurs changent les valeurs des attributs de l'objet formulaire	163
4.7.8.4	Custom de l'application	163
4.8	Module "Édition"	164
4.8.1	Introduction	165
4.8.2	Les états et lettres types	165
4.8.2.1	Paramétrer des états	165
4.8.2.2	Paramétrer des lettres-type	166
4.8.2.3	Actif, non actif	167
4.8.2.4	Les requêtes	167
4.8.2.4.1	Description	167
4.8.2.4.1.1	SQL	167
4.8.2.4.1.2	OBJET	167
4.8.2.4.2	Modèle de données	168
4.8.2.5	Les sous-états	168
4.8.2.6	Les champs de fusion	169
4.8.2.7	Les variables de remplacement	169
4.8.2.7.1	Les fichiers de configuration ../dyn/var*pdf.inc	169
4.8.2.7.2	Les méthodes globales de la classe du fichier ../obj/om_dbform.class.php	169
4.8.2.7.3	La table de paramètres om_parametre	169
4.8.2.8	Les logos	169
4.8.2.9	L'éditeur WYSIWYG	170
4.8.2.10	Les anciens fichiers de paramétrage	170
4.8.2.11	La prévisualisation	170
4.8.3	Les listings	170
4.8.3.1	Description de la fonctionnalité	170
4.8.3.2	Le fichier de paramétrage ../sql/pgsql/<OBJ>.pdf.inc.php	170
4.8.4	Les étiquettes	170
4.8.4.1	Description de la fonctionnalité	170
4.8.4.2	Le fichier de paramétrage ../sql/pgsql/<OBJ>.pdfetiquette.inc.php	170
4.8.5	Composants	170
4.9	Module "Import"	171
4.9.1	Principe	171
4.9.2	Paramétrage	171
4.9.2.1	Le script de paramétrage ../sql/pgsql/<OBJ>.import.inc.php	171
4.9.3	Composants	172
4.10	Module "Reqmo"	172
4.10.1	Principe	173
4.10.2	Paramétrage	173
4.10.2.1	Le script de paramétrage ../sql/pgsql/<OBJ>.reqmo.inc.php	173
4.10.3	Composants	174
4.11	Modules Géographiques	174
4.11.1	Module om_sig	174
4.11.1.1	la nouvelle architecture	174

4.11.1.1.1	tab_sig.php	175
4.11.1.1.2	form_sig.php	175
4.11.1.1.3	spg/map_compute_geom.php	176
4.11.1.1.4	spg/map_get_filters.php	176
4.11.1.1.5	spg/map_get_geojson_cart.php	177
4.11.1.1.6	spg/map_get_geojson_markers.php	177
4.11.1.1.7	spg/map_redirection_onglet.php	177
4.11.1.1.8	scr/export_sig.php	178
4.11.1.1.9	scr/requeteur.php	178
4.11.1.1.10	Nouvelles images dans img/ et nouvelle css pour l'interface om_sig	178
4.11.1.2	La classe om_map.class.php	178
4.11.1.2.1	Les variables globales	179
4.11.1.2.2	Les methodes	181
4.11.1.3	Le java script	182
4.11.1.4	Integration de la version 4.4.x en 4.4.5 :	186
4.11.1.5	Reprise des evolutions de la 4.4.0	187
4.11.1.5.1	openCadaastre	187
4.11.1.5.2	openAdresse	187
4.11.2	Module d'adresse	187
4.12	Abstraction du "layout" (ergonomie)	189
4.12.1	Le composant jquery	190
4.12.2	Les feuilles de style	190
4.13	Abstraction du "filestorage" (stockage des fichiers)	190
4.13.1	Principe général	191
4.13.2	Fonctionnement	191
4.13.2.1	Description de l'abstracteur	191
4.13.2.2	Description du fichier de configuration	191
4.13.2.3	Description des méthodes de la classe filestorage	192
4.13.2.4	Description du connecteur depredacted	193
4.13.2.5	Description du connecteur filesystem	193
4.13.2.6	Description du connecteur filetransferprotocol	194
4.13.2.7	Description du connecteur alfresco	194
4.13.3	Utilisation	194
4.13.3.1	Configuration du widget Upload	195
4.13.3.1.1	Contraintes	195
4.13.3.1.2	Métadonnées	195
4.13.3.2	Récupération du fichier	196
4.13.3.3	Scripts permettant de visualiser / d'accéder au fichier	196
4.13.3.3.1	spg/file.php	196
4.13.3.3.2	spg/voir.php	196
5	Tests et Intégration Continue	197
5.1	Pré-requis	197
5.1.1	PHPUnit	197
5.1.1.1	Installation	197
5.1.2	Selenium Server	197
5.1.2.1	Installation	197
5.1.3	Robot Framework	197
5.1.3.1	Installation	198
5.1.4	Jenkins	198
5.2	Arborescence du répertoire <i>tests</i>	198
5.2.1	<i>tests/om-tests</i>	198
5.2.2	<i>tests/config.xml</i>	199
5.2.3	<i>tests/000_generation.robot</i>	199

5.2.4	<i>tests/000_test_unitaire.php</i>	199
5.2.5	<i>tests/doc/</i>	200
5.2.6	<i>tests/results/</i>	200
5.2.7	<i>tests/binary_files/</i>	200
5.2.8	<i>tests/resources/</i>	200
5.2.9	<i>tests/resources/__init__.py</i>	200
5.2.10	<i>tests/resources/resources.robot</i>	200
5.2.11	<i>tests/resources/app/</i>	201
5.2.12	<i>tests/resources/app/gen/</i>	201
5.2.13	<i>tests/resources/app/__init__.py</i>	201
5.2.14	<i>tests/resources/app/om_tests.py</i>	201
5.2.15	<i>tests/resources/app/keywords.robot</i>	201
5.2.16	<i>tests/resources/core/</i>	202
5.3	Fonctionnement et Utilisation	202
5.3.1	Pré-requis	202
5.3.2	Tous les tests	202
5.3.3	Un seul TestSuite	203
5.4	Développement et bonnes pratiques	203
5.4.1	Documentation RobotFramework	203
5.4.2	Convention de nommage	203
5.4.3	Génération	204
5.4.4	Bonnes pratiques	204
6	Historique & Mises à niveau	205
6.1	La version 4.5	205
6.1.1	Les nouveautés de la version 4.5	205
6.1.2	Mettre à niveau depuis openMairie 4.4 vers 4.5	205
6.2	La version 4.4	206
6.2.1	Les nouveautés de la version 4.4	206
6.2.2	Mettre à niveau depuis openMairie 4.3 vers 4.4	206
6.2.2.1	Étape 1 : mettre à jour la base de données	206
6.2.2.1.1	La structure	206
6.2.2.2	Étape 2 : mise à jour du menu	207
6.2.2.3	Étape 3 : vérification des requêtes et logos	207
6.2.2.4	Étape 4 : système de stockage des fichiers	207
6.2.2.5	Les erreurs connues	207
6.3	La version 4.3	207
6.3.1	Les nouveautés de la version 4.3	207
6.3.2	Mettre à niveau depuis openMairie 4.2 vers 4.3	207
6.3.2.1	Étape 1 : mettre à jour les surcharges du framework	208
6.3.2.1.1	Classe <code>application</code>	208
6.3.2.1.2	Classe <code>dbForm</code>	208
6.3.2.1.3	Classe <code>formulaire</code>	208
6.3.2.1.4	Classe <code>table</code>	208
6.3.2.2	Étape 2 : mettre à jour la base de données	209
6.3.2.2.1	La structure	209
6.3.2.2.2	Les tables métier	209
6.3.2.2.2.1	PRIMARY KEY	209
6.3.2.2.2.2	FOREIGN KEY (PostgreSQL)	209
6.3.2.2.2.3	UNIQUE	209
6.3.2.2.2.4	NOT NULL	210
6.3.2.3	Étape 3 : mettre à jour les fichiers de surcharge du répertoire <code>sql/</code>	210
6.3.2.3.1	Alias des tables étrangères	210
6.3.2.3.2	La clause <code>ORDER BY</code>	210

	6.3.2.3.3	Les actions du tableau	210
	6.3.2.3.3.1	Les actions d'openMairie	210
	6.3.2.3.3.2	Les actions personnalisées	212
	6.3.2.3.3.3	Définition de l'action	212
	6.3.2.3.3.4	Définition du mode d'affichage en sous-tableau	212
	6.3.2.3.3.5	Définition de l'ordre d'affichage	212
	6.3.2.3.3.6	Définition des droits d'affichage	213
	6.3.2.4	Les erreurs connues	213
6.4	La version 4.2		213
6.4.1	Les nouveautés de la version 4.2		213
6.4.2	Mettre à niveau depuis openMairie 4.1 vers 4.2		213
	6.4.2.1	EXTERNALS.txt	213
	6.4.2.2	Regenerer les tables avec genfull.php	214
	6.4.2.3	Modifier les paramètres dyn	214
	6.4.2.4	Dans obj/	214
	6.4.2.5	Evolution om_sig_point vers om_sig_map	214
7	Règles		215
7.1	Convention de codage		215
7.1.1	L'indentation du code		215
7.1.2	Longueur maximum d'une ligne		216
7.1.3	Encodage des fichiers		216
7.1.4	Tags PHP		216
7.1.5	Règles typographiques		216
	7.1.5.1	Accolades	216
	7.1.5.2	Espacement	216
	7.1.5.3	Indentation	217
	7.1.5.4	Nommage des classes, fonctions, méthodes et variables	218
	7.1.5.5	Expressions	218
7.1.6	HTML Valide et W3C		218
7.1.7	Les commentaires dans le code		218
	7.1.7.1	Quand et comment commenter son code ?	220
7.1.8	Images		220
7.2	Versionnage		220
7.3	Documentation		220
7.3.1	Quelques bonnes pratiques		221
	7.3.1.1	Les blocs de code PHP	221
7.4	Publication		221
7.4.1	La documentation		221
7.5	Stratégies de développement		221
7.5.1	SD01		222
	7.5.1.1	Principes	222
	7.5.1.2	TDD (Test-Driven Development)	222
	7.5.1.3	Scénario-type de développement d'une évolution	222
	7.5.1.3.1	Initialisation	222
	7.5.1.3.2	Rédactions des tests	222
	7.5.1.3.3	Documentation	222
	7.5.1.3.4	Implémentation	223
	7.5.1.3.5	Intégration	223
	7.5.1.3.6	Incorporation	223
	7.5.1.4	Scénario-type de correction de bug	223
	7.5.1.5	Règles de contribution	223
	7.5.1.5.1	Contributeur	223
	7.5.1.5.1.1	Qui ?	223

	7.5.1.5.1.2	Comment contribuer ?	224
	7.5.1.5.1.3	Checklist	224
	7.5.1.5.2	Développeur	224
	7.5.1.5.2.1	Qui ?	224
	7.5.1.5.3	Responsable de version	224
	7.5.1.6	Label	224
7.5.2	SD02		225
	7.5.2.1	Principes	225
	7.5.2.2	Label	225
7.5.3	SD03		225
	7.5.3.1	Principes	225
	7.5.3.2	Label	225
8	Outils		227
8.1	Apache Subversion (SVN)		227
	8.1.1	Pré-requis	227
	8.1.2	L'arborescence	227
	8.1.3	Les règles d'or	228
	8.1.4	Les commandes basiques à connaître	228
	8.1.5	Externals	228
	8.1.6	Keywords	229
	8.1.7	Les clients graphiques	229
	8.1.8	Tutoriaux	229
	8.1.8.1	Importer un nouveau projet	229
	8.1.8.2	Publier une nouvelle version	230
	8.1.8.3	svn utilisation	231
8.2	GIT		232
8.3	Meld		232
8.4	POEdit		233
	8.4.1	Spécification dans le code des chaînes à traduire	233
	8.4.2	Préparation des dossiers de locales	233
	8.4.3	Installation et configuration de POEdit	234
	8.4.3.1	Installation	234
	8.4.3.2	Gestion de plusieurs langues	234
	8.4.4	Configuration d'un projet dans POEdit	234
	8.4.5	Traduction des chaînes	234
8.5	Sphinx		234
8.6	Github.com		235
	8.6.1	Créer un projet	235
	8.6.2	Importer la documentation depuis un projet subversion de l'adullact	235
	8.6.3	Faire l'import initial d'un projet sphinx	236
	8.6.4	Contribuer à une documentation	236
8.7	readthedocs.org		236
	8.7.1	Importer un nouveau projet sur RTD	237
	8.7.2	Paramétrer une nouvelle version d'un projet existant	237
9	Contributeurs		239
	Index		241

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.

Ce document a pour but de guider les développeurs dans la mise en œuvre d'un projet openMairie.

Avec plus de 50 applications développées pour les collectivités locales accessibles sur le site <http://www.openmairie.org>, nous souhaitons au travers de ce guide, diffuser notre expérience auprès des collectivités et des acteurs économiques du libre qui les accompagnent.

C'est donc une méthode conçue au fur à mesure de nos développements que nous vous proposons de partager et toutes remarques sont les bienvenues, alors n'hésitez pas à nous en faire part sur le forum du framework : <https://communaute.openmairie.org/c/framework>.

Nous avons conçu openMairie pour fabriquer une maquette grâce au générateur le plus en amont possible en s'appuyant sur le modèle de données et en intégrant des composants standards du « monde libre ».

Cette maquette permet très rapidement d'engager un dialogue participatif avec les utilisateurs, de concentrer le développeur uniquement sur le « métier » et de faire valider par l'utilisateur les évolutions successives.

Si vous débutez, il est préférable de commencer par le tutoriel « créer une application » qui permet de prendre en main facilement le générateur et le framework openMairie en vous guidant pas à pas dans la mise en place d'une gestion de courrier.

Le manuel d'usage complète l'exemple ci-dessus en vous décrivant le fonctionnement du framework : ergonomie, administration, génération, intégration, ...

Le manuel de référence s'adresse plus aux utilisateurs confirmés en décrivant le paramétrage, les classes formulaires et éditions du framework. Il a pour but de vous informer de manière complète sur le fonctionnement du framework.

Un chapitre particulier est dédié aux tests et à l'intégration continue. Le framework openMairie possède des tests unitaires et fonctionnels joués intégralement à chaque modification du code source afin d'assurer sa stabilité et sa pérennité.

Un chapitre est consacré à l'historique des versions du framework et aux conseils pour les mises à niveau.

Ce document rassemble enfin toutes les règles de codage du projet openMairie, ainsi que des outils pour aider et guider les développeurs de la communauté. Les règles indiquées doivent être appliquées pour qu'un projet puisse intégrer la distribution openMairie car l'objectif est de faciliter la lisibilité et la maintenance du code ainsi que la prise en main par les collectivités.

Toutes les dernières infos sur le framework sont disponibles ici : <http://www.openmairie.org/framework>. Bonne lecture !

CHAPITRE 1

Prérequis et installation

Sommaire

- *Prérequis et installation*
 - *Pré-requis*
 - *Installation*
 - *Installation des fichiers du framework*
 - *Télécharger l'archive zip*
 - *Décompresser l'archive zip dans le répertoire de votre serveur web*
 - *Création et initialisation de la base de données*
 - *Créer la base de données*
 - *Initialiser la base de données*
 - *Configuration de l'applicatif*
 - *Création des répertoires dédiés au stockage des données de l'applicatif*
 - *Positionner les permissions nécessaires au serveur web*
 - *Configuration de la connexion à la base de données*
 - *Connexion au framework*
 - *Ouverture dans le navigateur*
 - *Login*
 - *En cas d'erreur*
 - *Activer le mode debug*

1.1 Pré-requis

Vous devez avoir installé :

- un serveur web (apache, ...)
- PHP (Versions testées : 5.5 > 5.6)
- le moteur de base de données PostGreSQL (Versions testées : 9.1 > 9.4) avec l'extension PostGIS (Versions testées : 2.0 > 2.1)
- la librairie XML : libxml > 2.9.0

Sous windows, il est facile de trouver de la documentation pour l'installation de ces éléments en utilisant wamp (<http://www.wampserver.com/>) par exemple.

Sous Linux, il est facile de trouver de la documentation pour l'installation de ces éléments sur votre distribution.

Il est nécessaire de configurer l'installation PHP via le script php.ini comme suit pour cacher les erreurs « deprecated » liées à la librairie DBPEAR

```
error_reporting = E_ALL & ~E_DEPRECATED & ~E_STRICT
display_errors = On

avec la version php7 (ubuntu 16-04)

error_reporting = E_ALL & ~E_DEPRECATED & ~E_STRICT & ~E_WARNING & ~E_NOTICE
display_errors = On
```

1.2 Installation

1.2.1 Installation des fichiers du framework

1.2.1.1 Télécharger l'archive zip

http://adullact.net/frs/?group_id=265

1.2.1.2 Décompresser l'archive zip dans le répertoire de votre serveur web

- Exemple sous windows dans wamp : wamp/www/framework-openmairie
- Exemple sous linux avec debian : /var/www/framework-openmairie

1.2.2 Création et initialisation de la base de données

1.2.2.1 Créer la base de données

Il faut créer la base de données dans l'encodage UTF8. Par défaut la base de données s'appelle openexemple.

Dans un environnement debian :

```
createdb openexemple
```

1.2.2.2 Initialiser la base de données

Il faut initialiser les tables, les séquences et données de paramétrage grâce au script data/pgsql/install.sql

Dans un environnement debian depuis le répertoire data/pgsql/ :

```
psql openexemple -f install.sql
```

1.2.3 Configuration de l'applcatif

1.2.3.1 Création des répertoires dédiés au stockage des données de l'applcatif

Il est nécessaire de créer les répertoires suivants pour gérer le stockage des fichiers de logs, des fichiers temporaires et des fichiers stockés. L'arborescence à créer est décrite ci-après : *var/*.

1.2.3.2 Positionner les permissions nécessaires au serveur web

Dans un environnement debian :

```
chown -R www-data:www-data /var/www/framework-openmairie
```

1.2.3.3 Configuration de la connexion à la base de données

La configuration se fait dans le fichier *dyn/database.inc.php* :

```
<?php
...
// PostgreSQL
$conn[1] = array(
    "Framework openMairie", // Titre
    "pgsql", // Type de base
    "pgsql", // Type de base
    "postgres", // Login
    "postgres", // Mot de passe
    "tcp", // Protocole de connexion
    "localhost", // Nom d'hôte
    "5432", // Port du serveur
    "", // Socket
    "openexemple", // nom de la base
    "AAAA-MM-JJ", // Format de la date
    "openexemple", // Nom du schéma
    "", // Préfixe
    null, // Paramétrage pour l'annuaire LDAP
    null, // Paramétrage pour le serveur de mail
    null, // Paramétrage pour le stockage des fichiers
);
...
?>
```

1.3 Connexion au framework

1.3.1 Ouverture dans le navigateur

<http://localhost/framework-openmairie/>

“localhost” peut être remplacé par l’ip ou le nom de domaine du serveur.

1.3.2 Login

— Utilisateur « administrateur » :

- identifiant : admin
- mot de passe : admin

Le message de bienvenue doit être affiché « Votre session est maintenant ouverte. »

1.4 En cas d'erreur

1.4.1 Activer le mode debug

Il est possible d'activer le mode debug pour visualiser les messages d'erreur détaillés. Dans le fichier *dyn/debug.inc.php*, il faut commenter le mode production et décommenter le mode debug.

Mode production :

```
//define('DEBUG', VERBOSE_MODE);  
//define('DEBUG', DEBUG_MODE);  
define('DEBUG', PRODUCTION_MODE);
```

Mode debug :

```
//define('DEBUG', VERBOSE_MODE);  
define('DEBUG', DEBUG_MODE);  
//define('DEBUG', PRODUCTION_MODE);
```

Tutoriel - Créer une application

Ce chapitre vous propose de créer une application de gestion de courrier pas à pas.

2.1 Créer la base de données

Vous devez au préalable récupérer le framework. Dans le répertoire www de votre serveur apache :

```
svn checkout svn://scm.adullact.net/scmrepos/svn/openmairie/openmairie_exemple/trunk_
↪openExemple
```

Il vous est proposé de créer la base de données sous PostgreSQL :

- Créer les tables nécessaires au framework openMairie :

```
cd data/pgsql
sudo su postgres
dropdb openexemple && createdb openexemple && psql openexemple -f install.sql
```

- Créer les tables nécessaires à notre exemple :

- table courrier

courrier	int 8	cle primaire
dateenvoi	date	
objetcourrier	text	
emetteur	int8	cle secondaire
service	int8	cle secondaire

- table emetteur

emetteur	int 8	cle primaire
nom	varchar 20	
prenom	varchar 20	

- table service

service	int 8	cle primaire
libelle	varchar 20	

— La requête correspondante en PostgreSQL est la suivante :

```
-- Selection du schéma

SET search_path TO openexemple;

-- Création des séquences

CREATE SEQUENCE emetteur_seq
START WITH 1
INCREMENT BY 1
NO MINVALUE
NO MAXVALUE
CACHE 1;

CREATE SEQUENCE service_seq
START WITH 1
INCREMENT BY 1
NO MINVALUE
NO MAXVALUE
CACHE 1;

CREATE SEQUENCE courrier_seq
START WITH 1
INCREMENT BY 1
NO MINVALUE
NO MAXVALUE
CACHE 1;

-- Création des tables

CREATE TABLE emetteur (
    emetteur      int PRIMARY KEY, -- clé primaire
    nom           varchar(20),
    prenom        varchar(20)
);

CREATE TABLE service (
    service       int PRIMARY KEY, -- clé primaire
    libelle       varchar(20)
);

CREATE TABLE courrier (
    courrier      int PRIMARY KEY,          -- clé primaire
    dateenvoi    date,
    objetcourrier text,
    emetteur      int REFERENCES emetteur,  -- clé étrangère
    service       int REFERENCES service    -- clé étrangère
);
```

— Modifier le paramétrage openMairie pour faire un accès à la base créée :

dyn/database.inc.php

voir *[Paramétrage de la connexion à la base de données](#)*

— Accéder avec votre navigateur sur openExemple :

login : **admin**

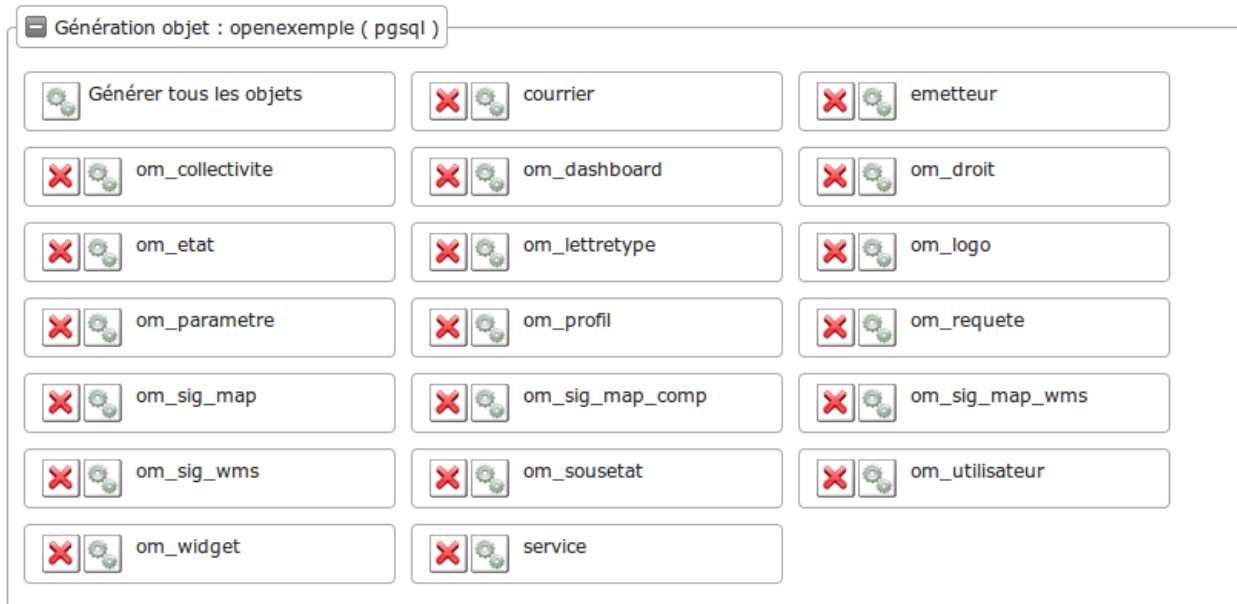
mot de passe : **admin**

2.2 Créer les formulaires

Nous allons maintenant créer les formulaires à l'aide du générateur.

Pour cela, il faut aller dans le menu **Administration -> Générateur**.

Vous devez avoir 3 nouveaux boutons : courrier, service et emetteur.



Avant de commencer, l'utilisateur apache **www-data** doit avoir les droits d'écriture dans les répertoires */gen* , */sql* et */obj*.

2.2.1 Générer les formulaires et édition du courrier

En appuyant sur le bouton de courrier, vous avez les choix de génération :

Choix des fichiers à générer

Selection	Nom Fichier	Generer
formulaire		
☒	courrier.inc.php	../gen/sql/pgsql/courrier.inc.php [Le fichier n'existe pas ou n'est pas accessible]
☒	courrier.inc.php	../sql/pgsql/courrier.inc.php [Le fichier n'existe pas ou n'est pas accessible]
☒	courrier.form.inc.php	../gen/sql/pgsql/courrier.form.inc.php [Le fichier n'existe pas ou n'est pas accessible]
☒	courrier.form.inc.php	../sql/pgsql/courrier.form.inc.php [Le fichier n'existe pas ou n'est pas accessible]
☒	courrier.class.php	../gen/obj/courrier.class.php [Le fichier n'existe pas ou n'est pas accessible]
☒	courrier.class.php	../obj/courrier.class.php [Le fichier n'existe pas ou n'est pas accessible]
édition		
☐	courrier.pdf.inc.php	../sql/pgsql/courrier.pdf.inc.php [Le fichier n'existe pas ou n'est pas accessible]
reqmo		
☐	courrier.reqmo.inc.php	../sql/pgsql/courrier.reqmo.inc.php [Le fichier n'existe pas ou n'est pas accessible]
☐	courrier_emetteur.reqmo.inc.php	../sql/pgsql/courrier_emetteur.reqmo.inc.php [Le fichier n'existe pas ou n'est pas accessible]
☐	courrier_service.reqmo.inc.php	../sql/pgsql/courrier_service.reqmo.inc.php [Le fichier n'existe pas ou n'est pas accessible]
divers		
☐	courrier.import.inc.php	../sql/pgsql/courrier.import.inc.php [Le fichier n'existe pas ou n'est pas accessible]

Générer les fichiers de la table : 'courrier' [Retour](#)

Au préalable, le générateur a fait une analyse de la base de données :

analyse

Analyse de la base de données pgsql ➔ openexemple.openexemple

Elements	Infos
Tables de la base de données	[om_lettretype] [om_profil] [om_collectivite] [om_sousetat] [om_logo] [om_utilisateur] [om_parametre] [om_dashboard] [om_sig_map_comp] [emetteur] [om_droit] [service] [om_sig_map_wms] [om_requete] [om_sig_wms] [om_widget] [om_sig_map] [om_etat]
Table : courrier	[clé N - clé automatique] [courrier] [longueur enregistrement : 65]
Champs	[courrier 11 int] [dateenvoi 12 date] [objetcourrier -5 blob] [emetteur 11 int] [service 11 int] [registre 20 string]
Sous-formulaire	
Clé secondaire	[emetteur] [service]

Le générateur a donc détecté 2 clés secondaires et aucun sous-formulaire.

C'est pour cela qu'il propose 3 « reqmos » : 1 « reqmo » global et 2 « reqmos » suivant les clés secondaires.

Par défaut, seules les options du formulaire sont cochées.

Si vous le refaites plus tard, seules celles fabriquées par le générateur seront cochées.

Cochez les toutes :

☒	courrier.inc.php
☒	courrier.inc.php
☒	courrier.form.inc.php
☒	courrier.form.inc.php
☒	courrier.class.php
☒	courrier.class.php
☒	courrier.pdf.inc.php
☒	courrier.reqmo.inc.php
☒	courrier_emetteur.reqmo.inc.php
☒	courrier_service.reqmo.inc.php
☒	courrier.import.inc.php

En cliquant sur valider, vous avez le message :


Table : courrier

Aucun changement de ../gen/sql/pgsql/courrier.inc.php
Génération de ../sql/pgsql/courrier.inc.php
Aucun changement de ../gen/sql/pgsql/courrier.form.inc.php
Aucun changement de ../sql/pgsql/courrier.form.inc.php
Génération de ../gen/obj/courrier.class.php
Génération de ../obj/courrier.class.php
->affichage colone ok 4,3076923076923 >= 2.5
Génération de ../sql/pgsql/courrier.pdf.inc.php
Génération de ../sql/pgsql/courrier.reqmo.inc.php
Génération de ../sql/pgsql/courrier_emetteur.reqmo.inc.php
Génération de ../sql/pgsql/courrier_service.reqmo.inc.php
Génération de ../sql/pgsql/courrier.Import.inc.php

Le paramétrage utilisé est le paramétrage standard.

Vous pouvez le modifier : voir *générateur/paramétrage*.

L'affichage par colone est « ok », ce qui veut dire que la taille des colones dans le fichier pdf sera complet (attention le script ne prend pas le champ blob).

2.2.2 Générer les formulaires et édition de l'émetteur

Nous allons procéder de la même manière avec le bouton émetteur.

L'analyse de la base de données est la suivante :

analyse

Analyse de la base de données pgsql ➔ openexemple.openexemple

Elements	Infos
Tables de la base de données	[om_lettretype] [om_profil] [om_collectivite] [om_sousetat] [om_logo] [om_utilisateur] [om_parametre] [om_dashboard] [om_sig_map_comp] [courrier] [om_droit] [service] [om_sig_map_wms] [om_requete] [om_sig_wms] [om_widget] [om_sig_map] [om_etat]
Table : emetteur	[clé N - clé automatique] [emetteur] [longueur enregistrement : 136]
Champs	[emetteur 11 int] [nom 20 string] [prenom 20 string] [adresse 40 string] [cp 5 string] [ville 40 string]
Sous-formulaire	[courrier]
Clé secondaire	

Le générateur repère un sous formulaire courrier. Effectivement, il y a une relation de un à plusieurs entre émetteur et courrier : un émetteur peut avoir 0 à plusieurs courriers.

En cliquant sur toutes les options puis en validant, vous avez le message suivant :

i Table : emetteur

Aucun changement de ../gen/sql/pgsql/emetteur.inc.php
 Aucun changement de ../sql/pgsql/emetteur.inc.php
 Aucun changement de ../gen/sql/pgsql/emetteur.form.inc.php
 Aucun changement de ../sql/pgsql/emetteur.form.inc.php
 Aucun changement de ../gen/obj/emetteur.class.php
Génération de ../obj/emetteur.class.php
 ->affichage colone incomplet 2,0588235294118 < 2.5
Génération de ../sql/pgsql/emetteur.pdf.inc.php
Génération de ../sql/pgsql/emetteur.reqmo.inc.php
Génération de ../sql/pgsql/emetteur.import.inc.php

2.2.3 Générer les formulaires et édition de service

Nous allons procéder de la même manière avec le bouton service

L'analyse de la base de données est la suivante :

analyse

Analyse de la base de données pgsql ➔ openexemple.openexemple

Elements	Infos
Tables de la base de données	[om_lettretype] [om_profil] [om_collectivite] [om_sousetat] [om_logo] [om_utilisateur] [om_parametre] [om_dashboard] [om_sig_map_comp] [emetteur] [courrier] [om_droit] [om_sig_map_wms] [om_requete] [om_sig_wms] [om_widget] [om_sig_map] [om_etat]
Table : service	[clé N - clé automatique] [service] [longueur enregistrement : 31]
Champs	[service 11 int] [libelle 20 string]
Sous-formulaire	[courrier]
Clé secondaire	

Le générateur repère un sous formulaire courrier. Effectivement, il y a une relation de un à plusieurs entre service et courrier : un service peut avoir 0 à plusieurs courriers.

En cliquant sur toutes les options, vous avez le message suivant :

Table : service

Génération de ../gen/sql/pgsql/service.inc.php
 Aucun changement de ../sql/pgsql/service.inc.php
 Aucun changement de ../gen/sql/pgsql/service.form.inc.php
 Aucun changement de ../sql/pgsql/service.form.inc.php
 Aucun changement de ../gen/obj/service.class.php
 Aucun changement de ../obj/service.class.php
 ->affichage colone ok 9,0322580645161 >= 2.5
 Aucun changement de ../sql/pgsql/service.pdf.inc.php
 Aucun changement de ../sql/pgsql/service.reqmo.inc.php
 Aucun changement de ../sql/pgsql/service.import.inc.php

2.2.4 Intégrer les formulaires dans le menu

Pour accéder à nos formulaires, nous allons les intégrer dans le menu (*voir framework/parametrage/menu gauche*).

Nous allons appeler le formulaire depuis le menu :

- option **Application** -> tab.php?obj=courrier
- option **Paramétrage** -> tab.php?obj=emetteur
- option **Paramétrage** -> tab.php?obj=service

Il faut ouvrir avec un éditeur le fichier *dyn/menu.inc.php* et insérer le code suivant :

```
// {{{ Rubrique APPLICATION

$links[] = array(
    "href" => "../scr/tab.php?obj=courrier",
    "class" => "courrier",
    "title" => _("courrier"),
    "right" => "courrier"
```

(suite sur la page suivante)

(suite de la page précédente)

```
);

// {{{ Rubrique PARAMETRAGE

$links[] = array(
    "href" => "../scr/tab.php?obj=emetteur",
    "class" => "emetteur",
    "title" => _("emetteur"),
    "right" => "emetteur"
);

$links[] = array(
    "href" => "../scr/tab.php?obj=service",
    "class" => "service",
    "title" => _("service"),
    "right" => "service"
);
```

Il faut également bien placer le code, c'est à dire dans la bonne rubrique (précisée en commentaire) après

```
$links = array();
```

et avant

```
$rubrik['links'] = $links;
```

Enfin pour y accéder il faut soit donner les droits via le menu framework, soit (et c'est en l'occurrence le cas) dans le fichier config.inc.php (option utilisée que pour le développement) décommenter la ligne

```
$config['permission_if_right_does_not_exist'] = true;
```

Vous pouvez maintenant accéder à vos formulaires par le menu.

2.2.5 Menu

Application -> Courrier

Cette opération affiche la table courrier :

On accède en appuyant sur + au formulaire d'insertion où les champs sont :

- la date du courrier avec calendrier,
- l'objet du courrier dans un champ textarea,
- deux contrôles « select » pour le service et l'emetteur.

Paramétrage -> Emetteur

Cette operation affiche la table emetteur :

En appuyant sur +, on accède à la saisie.

L'onglet courrier est inactif tant que l'émetteur n'est pas saisi et validé.

Paramétrage -> Service

Cette opération affiche la table service :

En appuyant sur +, on accède à la saisie.

L'onglet courrier est inactif tant que le service n'est pas saisi.

Vous pouvez accéder aux éditions et requêtes mémorisées :

Export -> Edition

Cet option affiche l'ensemble des éditions pdf :

pour en savoir plus voir [framework/edition](#)

Export -> Requêtes Mémorisées

Cette option affiche les requêtes mémorisées :

Export ➔ Requêtes mémorisées

Les requêtes mémorisées permettent d'exporter des données de la base-de-données pour une utilisation externe à l'application. Veuillez cliquer sur l'objet à exporter pour accéder à un formulaire vous permettant de choisir les paramètres de l'export.

☐ Choix de la requête mémorisée

➔ courrier

➔ courrier_emetteur

➔ courrier_service

➔ emetteur

➔ collectivité

➔ droit

➔ paramètre

➔ Paramètres d'une collectivité

➔ profil

➔ utilisateur

➔ service

openExemple Version 4.4.0-dev | Documentation | openMairie.org

pour en savoir plus voir *framework/reqmo*

Vous pouvez accéder aux éditions en appuyant dans le formulaire d'affichage sur l'imprimante.

Vous pouvez accéder au fichiers d'import :

Administration -> Import

Cette option affiche les scripts d'imports :

Import

Cette page vous permet d'importer des données au format CSV directement dans la base de données.

☐ Choix de la table d'import :

Table

☐ Import du fichier CSV :

Fichier   

Séparateur

Importer le fichier dans la table : 'courrier'  **Retour**

openExemple Version 4.4.0-dev | Documentation | openMairie.org

Pour en savoir plus voir *framework/import*.

2.3 Personnaliser son application

Nous allons maintenant personnaliser notre application.

Pour ce faire, nous allons saisir un jeu de données.

Vous pouvez le faire avec les formulaires, l'incrémentation des séquences étant faite par le framework. Tout comme la création des tables stockant les séquences (méthode *setId* des objets métier).

Sinon exécutez la requête PostgreSQL suivante :

```
-- création des tables de séquence déjà faite

-- Selection du schéma

SET search_path TO openexemple;

-- insertion de deux émetteurs avec récupération et incrémentation de la table de_
↳séquences

INSERT INTO emetteur (emetteur, nom, prenom) VALUES
(nextval('emetteur_seq'), 'dupont', 'pierre'),
(nextval('emetteur_seq'), 'durant', 'jacques');

-- insertion de deux services avec récupération et incrémentation de la table de_
↳séquences

INSERT INTO service (service, libelle) VALUES
(nextval('service_seq'), 'informatique'),
(nextval('service_seq'), 'telephonie');

-- insertion de deux courriers avec récupération et incrémentation de la table de_
↳séquences

INSERT INTO courrier (courrier, dateenvoi, objetcourrier, emetteur, service) VALUES
(nextval('courrier_seq'), '2010-12-01', 'Proposition de fourniture de service', 1, 1),
(nextval('courrier_seq'), '2010-12-02', 'Envoi de devis pour formation openMairie', 2,
↳ 1);
```

2.3.1 Faire un affichage courrier plus convivial

L’affichage des courriers se fait avec des libellés générés automatiquement.

Ainsi dans le fichier *gen/sql/pgsql/courrier.inc.php* (qui est inclus dans le fichier *sql/pgsql/courrier.inc.php* que vous pourrez modifier) vous avez la variable **\$champAffiche**.

Vu que ce fichier a été créé par le générateur (et est en lecture seule) et vu que nous souhaitons modifier la variable (pour par exemple avoir le nom et le prénom de l’émetteur au lieu de simplement son nom) il nous faut ouvrir le fichier *sql/pgsql/courrier.inc.php* où nous allons (après l’*include* !) réaffecter à la variable **\$champAffiche** la valeur suivante :

```
$champAffiche = array(
    'courrier.courrier as "'.$_("courrier").'"',
    'to_char(courrier.dateenvoi ,\'DD/MM/YYYY\') as "'.$_("dateenvoi").'"',
    'concat(emetteur.nom,\'' \',emetteur.prenom) as "'.$_("emetteur").'"',
    'service.libelle as "'.$_("service").'"',
);
```

Il est possible que l’opération vous soit refusée (seul **www-data** ayant les droits d’écriture).

Si tel est le cas échéant il faudra se rajouter les permissions.

Le résultat est le suivant :

Courrier	Dateenvoi	Emetteur	Service
1	01/12/2010	dupont pierre	informatique
2	02/12/2010	durant jacques	informatique

De la même manière, toujours dans le même fichier, vous pouvez changer les options de la zone de recherche en

réaffectant la variable **\$champRecherche**. Actuellement on peut en plus de *Tous* faire une recherche sur *courrier*, *emetteur* et *service*.

C'est parce qu'à l'origine, dans le fichier généré, **\$champRecherche** avait été affectée comme telle :

```
$champRecherche = array(
'courrier.courrier as "'.$_("courrier").'",
'emetteur.nom as "'.$_("emetteur").'",
'service.libelle as "'.$_("service").'",
);
```

Supprimez un ou plusieurs élément(s) du tableau et il disparaîtra de la zone de recherche. Par exemple...

```
$champRecherche = array(
'emetteur.nom as "'.$_("emetteur").'",
);
```

... donnera :

service	registre
telephonie	2013-3
informatique	2013-1

Nous souhaitons maintenant avoir les derniers courriers au début de la page affichée. Nous n'avons pas besoin d'aller réécrire la requête, il existe une variable texte comprenant l'instruction de tri. Réaffectez cette variable **\$tri** dans votre *courrier.inc.php* de la manière suivante :

```
$tri= " order by dateenvoi desc";
```

Le résultat est le suivant :

dateenvoi	emetteur
02/12/2010	durant
01/12/2010	dupont

Pour en savoir plus sur ces variables voir *framework/affichage*.

2.3.2 Rendre obligatoire des champs

Nous avons affiché le courrier avec une jointure de type *LEFT JOIN* ce qui ne rend pas obligatoire la saisie de l'émetteur et du service (auquel le courrier est affecté).

Nous devons surcharger la méthode **verifier()**.

Dans *obj/courrier.class.php* la méthode à insérer après le constructeur est celle-ci :

```
function verifier($val,&$db,$DEBUG) {
    parent::verifier($val,$db,$DEBUG);
}
```

(suite sur la page suivante)

(suite de la page précédente)

```
// Les champs service et emetteur sont obligatoires
if ($this->valF['service']== "") {
    $this->correct=False;
    $this->addToMessage(_('service')."&nbsp;".$_('obligatoire')."&nbsp;!");

    //
}
if ($this->valF['emetteur']== "") {
    $this->correct=False;
    $this->addToMessage(_('emetteur')."&nbsp;".$_('obligatoire')."&nbsp;!");
}
}
```

Par défaut le premier champ (ici *dateenvoi*) est obligatoire, cette option est modifiable dans le générateur.

La commande *parent* : `:verifier($val,$db,$DEBUG)`; permet de ne pas neutraliser la fonction surchargée (ici dans `gen/obj/courrier.class.php`)

Pour plus d'informations voir *framework/methode*.

2.3.3 Valoriser un champ par défaut

Pour simplifier la saisie, nous souhaitons mettre la date du jour dans le champ *dateenvoi* lors d'un ajout de courrier.

Nous allons surcharger la methode `setVal()` dans *obj/courrier.class.php* de la manière suivante :

```
function setVal(&$form, $maj, $validation, &$db, $DEBUG=null){
    parent::setVal($form, $maj, $validation, $db, $DEBUG=null);

    if ($validation==0) {
        if ($maj == 0){
            $form->setVal("dateenvoi", date('Y-m-d'));
        }
    }
}
```

Le champ *dateenvoi* contiendra la date système (*date("Y_m-d")*) si la validation est égale à 0 (ce qui signifie que le formulaire n'a pas été validé) et si **\$maj** est égal à 0 (ce qui signifie qu'il s'agit d'un ajout).

Les autres valeurs que peut prendre **\$maj** sont :

- 1 : modifier
- 2 : supprimer
- 3 : consulter

2.3.4 Mettre en majuscule un champ

Nous souhaitons maintenant mettre en majuscule le champ *nom* de la table *emetteur*.

Nous allons surcharger la méthode `setOnChange()` dans *obj/emetteur.class.php* de la manière suivante :

```
function setOnChange(&$form, $maj) {
    parent::setOnChange($form, $maj);

    $form->setOnChange("nom", "this.value=this.value.toUpperCase()");
}
```

A la saisie ou à la modification du nom, le champ se mettra en majuscule.

2.3.5 Principe à retenir

Voilà quelques exemples des possibilités de modification dans les fichiers sql (répertoire *sql/...*) et dans les méthodes de l'objet (répertoire *obj/...*).

En aucun cas il ne faut modifier les fichiers dans *gen/* qui est l'espace de travail propre au générateur.

Nous allons dans le prochain chapitre modifier la base et régénérer les écrans sans mettre en danger votre personnalisation.

2.4 Modifier la base et régénérer

Le framework openMairie permet de modifier la base et prendre en compte ces modifications en régénérant les scripts **sans** mettre en péril la personnalisation que vous avez effectuée.

Nous vous proposons de rajouter un champ *registre* dans la table *courrier* et de rajouter l'adresse dans la table *emetteur*.

2.4.1 Rajouter un champ registre dans courrier

Il est proposé de rajouter un champ *registre* dans le courrier dont le but est de stocker le numéro de registre du courrier sous la forme *annee_numero_d_order*.

Nous allons d'abord créer un champ *registre* dans la table *courrier* de la manière suivante :

```
ALTER TABLE courrier ADD registre VARCHAR( 20 ) ;
```

Vous devez régénérer votre application courrier dans l'option du menu **Administration -> Générateur -> Courrier** et laisser cochées les options par défaut :

```
gen/obj/courrier.class.php
gen/sql/pgsql/courrier.inc.php
gen/sql/pgsql/courrier.form.inc.php
```

Validez l'opération.

Vous pouvez remarquer si vous allez sur le formulaire d'ajout qu'il y a un nouveau champ *registre*. Votre personnalisation n'est pas affectée.

Nous voulons que le numéro de registre se mette en ajout de manière automatique une fois le formulaire validé.

Il faut donc surcharger les méthodes suivantes dans *obj/courrier.class.php* :

```
// pour que registre ne soit pas modifiable

function setType(&$form,$maj) {
    parent::setType($form,$maj);
    $form->setType('registre', 'hiddenstatic');
}

// pour la mise à jour de la séquence avant l'ajout de l'enregistrement

function triggerajouter($id,&$db,$val,$DEBUG) {
    // prochain numero de registre
    // fonction DB pear
```

(suite sur la page suivante)

(suite de la page précédente)

```

$temp= $db->nextId("registre");
// fabrication du numero annee_no_d_ordre
$temp= date('Y')."-".$temp;
$this->valF['registre'] = $temp;
}

```

Si vous souhaitez que registre apparaisse dans l’affichage de la table, vous devez aussi modifier la variable *champAffiche* de *sql/pgsql/courrier.inc* de la manière suivante :

```

$champAffiche = array(
    'courrier.courrier as "'.$_("courrier").'"',
    'to_char(courrier.dateenvoi ,\'DD/MM/YYYY\') as "'.$_("dateenvoi").'"',
    'concat(emetteur.nom,\' \' ,emetteur.prenom) as "'.$_("emetteur").'"',
    'service.libelle as "'.$_("service").'"',
    'registre'
);

```

Votre affichage de la table courrier est modifié.

2.4.2 Rajouter l’adresse dans emetteur

Il est proposé de rajouter l’adresse de l’emetteur à savoir : le libellé, le code postal et la ville.

La requête est la suivante :

```

ALTER TABLE emetteur ADD adresse VARCHAR( 40 ) ,
ADD cp VARCHAR( 5 ) ,
ADD ville VARCHAR( 40 ) ;

```

Vous devez régénérer votre application courrier en allant dans l’option du menu : administration -> generateur -> emetteur et laisser cochées les options par défaut :

```

gen/obj/emetteur.class.php
gen/sql/pgsql/emetteur.inc.php
gen/sql/pgsql/emetteur.form.inc.php

```

Validez l’opération.

N’ayant pas modifié *sql/pgsql/emetteur.inc*, le framework fonctionne avec le code généré.

2.4.3 Améliorer la présentation du formulaire emetteur

Nous pouvons continuer à améliorer les présentations de nos formulaires en utilisant les méthodes *setGroupe()* et *setRegroupe()* dans le script *obj/emetteur.class.php*.

Il vous est proposé d’insérer dans votre script *obj/emetteur.class.php* le code suivant :

```

function setLayout(&$form, $maj) {

    $form->setFieldset('nom','D',_('nom'),'collapsible');
    $form->setFieldset('prenom','F');

    $form->setFieldset('adresse','D',_('adresse'),'startClosed');
    $form->setFieldset('ville','F');
}

```

Le fieldset nom est affiché par défaut, pas celui de l'adresse :

The screenshot shows a web form with a title 'emetteur 1'. Below the title is a form container. Inside, there is a 'nom' field with a minus icon and the value 'nom dupont'. To its right is a 'prenom' field with the value 'prenom pierre'. Below these is an 'adresse' field with a plus icon, which is currently collapsed.

Vos formulaires sont maintenant au point.

Le paragraphe suivant vous indique les surcharges d'openCimetiere que vous pouvez intégrer dans votre exemple, maintenant que vous avez la méthode.

2.4.4 Les surcharges d'openCourrier

Vous pouvez utiliser openCourrier qui est téléchargeable au lien suivant :

http://www.adullact.net/frs/?group_id=297

Si les surcharges qui ont été faites dans notre exemple sont celles d'openCourrier, il y a d'autres surcharges dans le script *courrier.class.php* d'openCimetiere :

Les méthodes setLib, setGroupe et setRegroupe permettent **une présentation en fieldset** du courrier (utilisation des champs vide 1 à 5 voir *sql/pgsql/courrier.form.inc*).

Il y a d'autres objets métier qui ont des surcharges intéressantes, par exemple l'objet *obj/dossier.class.php* où vous avez un upload pour télécharger des fichiers.

Vous pouvez regarder également l'application openCourrier mais attention à la base de données qui est en MySQL :

- openCourrier fonctionne avec des restrictions d'accès par service et les méthodes de login ont été modifiées dans *obj/utills.class.php* ainsi qu'*utilisateur.class.php* qui a dans openCourrier un champ service.
- l'objet *obj/tachenonsolde.class.php* est un exemple de surcharge de *tache.class.php* qui affiche que les tâches non soldées
- vous pouvez aussi regarder deux scripts de traitement :
 - *trt/num_registre.php* qui remet à 0 le numéro de registre
 - *trt/archivage.php* qui transfère en archive les courriers avant une date

Vous avez également des détails sur les traitements dans le chapitre *framework/util* notamment sur la mise à jour du registre.

2.5 Créer ses états

Il vous est proposé de créer un état des courriers par service.

Il sera utilisé dans ce chapitre l'assistant état et sous-état du générateur.

Quittez le projet openCimetiere et revenez à openExemple.

2.5.1 Créer l'état service

Nous allons utiliser l'assistant état du générateur dans le menu :

Administration -> Générateur -> Assistants

Choisir *Création d'état* puis choisir dans le select l'option service.

Ensuite avec la touche *CTRL* sélectionner les champs *service.service* et *service.libellé*.

Cliquer ensuite sur *Import service dans la base*.

Administration ➔ Générateur ➔ état

cet assistant vous permet de créer des états directement à partir de vos tables

☐ Choix table :

fichier

☐ choix des champs

Utilisez ctrl key pour choix multiple

service.service
service.libellé

Import service dans la base

openExemple Version 4.4.0-dev | Documentation | openMairie.org

Un message apparaît : *service enregistré*.

Vous avez créé un enregistrement qui a pour identifiant *service* dans la table *om_etat*.

SELECT * FROM "om_etat" LIMIT 50 [Modifier](#)

☐ modifier	om_etat	om_collectivite	id	libelle	actif
☐	1	1	om_collectivite	om_collectivite gen le 12/11/2010	t
☐	2	1	service	service gen le 22/11/2013	f

Vous devez rendre d'abord votre état *service* *actif* pour pouvoir y accéder.

Il faut maintenant permettre l'accès dans l'affichage du service :

- Ouvrir le fichier *sql/pgsql/service.inc.php*
- Ajouter le script suivant :

```
$href[3] = array(
    "lien" => "../pdf/pdfetat.php?obj=".$obj."&idx=",
    "id" => "",
    "lib" => "<img src='../om-theme/img/pdf-16x16.png' alt='\"
    ._("Edition PDF").\" title='\"
    ._("Edition PDF").\" />",
);
```

Nous rajoutons la ligne 3 dans le tableau href. Vous avez un état lié à l'affichage du service.

Il y a des exemples d'utilisation de href dans *om_collectivité*, *om_etat*, *om_utilisateur*,...

2.5.2 Créer le sous-état courrier

Nous allons utiliser l'assistant sous-état du générateur dans le menu :

Administration -> Générateur -> Assistants -> Création sous-état

Nous choisissons la table courrier et nous surlignons les champs *courrier.dateenvoi*, *courrier.objetcourrier*, *courrier.emetteur* et *courrier.registre*.

Nous choisissons *courrier.service* comme clé secondaire pour faire le lien avec service.

Administration > Générateur > Sous-état

cet assistant vous permet de créer des sous-états directement à partir de vos tables

☐ Choix table :

fichier

☐ choix des champs

Utilisez ctrl key pour choix multiple

choisir la clé de sélection

openExemple Version 4.4.0-dev | Documentation | openMairie.org

En cliquant sur *Import courrier dans la base* vous créez un enregistrement ayant pour identifiant *courrier.service* dans la table *om_sousetat* :

SELECT * FROM "om_sousetat" LIMIT 50 [Modifier](#)

☐ modifier	om_sousetat	om_collectivite	id	libelle	actif
☐	1	1	om_parametre.om_collectivite	gen le 12/11/2010	t
☐	2	1	courrier.service	gen le 22/11/2013	f

2.5.3 Associer le sous-état *courrier* à l'état *service*

Vous devez rendre d'abord votre sous-état *courrier.service* actif pour pouvoir l'associer.

Allez dans l'option **Sous Etat** du menu **Paramétrage**.

Recherchez le sous-état *courrier.service* et modifiez le en cochant actif (premier fieldset : *collectivité*).

Il vous faut maintenant associer le sous-état *courrier.service* à l'état *service*.

Allez dans l'option **Etat** du menu **Paramétrage**.

Modifiez l'état *service* et dans le fieldset à déplier *Sous-état(s)* après l'avoir coché actif sélectionnez le sous-état *courrier.service*.

Attention la mise en place de sous état dans un état a été modifié dans la version 4.5 du framework La version ci dessous est om 4.4.

Sous-état(s)

sous-état

se_font *

se_margeleft *

se_margetop *

se_margeright *

se_couleurtexte *

Choisir sous-état

om_parametre.om_collectivite gen le 12/11/2010

courrier.service gen le 22/11/2013

helvetica

8

5

5

#000000

Vous avez désormais un état des courriers par service (**Paramétrage -> Service -> Edition PDF**) :

application → service

service

1 - 2 enregistrement(s) sur 2

Tous

Recherche

+

service

libellé

1 informatique

2 telephonie

openExemple Version 4.4.0-dev | Documentation | openMairie.org



le 22/11/2013

1

Informatique

liste courrier

DATEENVOI	OBJETCOURRIER	EMETTEUR	REGISTRE
2013-11-21	Proposition de fournitures de service.	1	2013-1
2013-11-21	Envoi de devis pour formation openMairie.	1	2013-2

2.5.4 Mettre le nom et le prénom de l'émetteur dans le sous-état

Nous souhaitons mettre le nom et le prénom de l'émetteur à la place de la clé secondaire.

Vous devez modifier la requête sql du sous-état *courrier.service* dans la table *om_sousetat* de la manière suivante :

```
select courrier.dateenvoi as dateenvoi,
courrier.objetcourrier as objetcourrier,
concat(emetteur.nom, ' ', emetteur.prenom) as emetteur,
courrier.registre as registre
from &DB_PREFIXEcourrier LEFT JOIN &DB_PREFIXEmetteur on emetteur.emetteur =
↳courrier.emetteur
where courrier.service='&idx'
```

Votre nouvel état a la forme suivante :



le 22/11/2013

1

informatique

liste courrier

DATEENVOI	OBJETCOURRIER	EMETTEUR	REGISTRE
2013-11-21	Envoi de devis pour formation openMairie.	dupont pierre	2013-2
2013-11-21	Proposition de fournitures de service.	dupont pierre	2013-1

Pour intégrer la date du jour, il faut créer un fichier varetatpdf.inc dans le répertoire /dyn dans lequel on va mettre le code suivant :

```
<php
// aujourd'hui
$titre=str_replace("&aujourd'hui",date('d/m/Y'),$titre);
$corps=str_replace("&aujourd'hui",date('d/m/Y'),$corps);

?>
```

Si on souhaite intégrer les paramètres de la table om_paramètre, on rajoute dans le même fichier le code suivant :

```
// Remplacement des paramètres présents dans le tableau '$collectivite'
// qui contient l'intégralité de la table 'om_parametre'
if (isset($collectivite) && $collectivite != array()) {
    //
    foreach (array_keys($collectivite) as $elem) {
        //
        $temp = "&".$elem;
        $collectivite_elem = $collectivite[$elem];
        ////
        //if (DBCHARSET == 'UTF8') {
        //    //
        //    $collectivite_elem = utf8_decode($collectivite_elem);
        //}
        //
        $titre = str_replace($temp, $collectivite_elem, $titre);
        $corps = str_replace($temp, $collectivite_elem, $corps);
        $sql = str_replace($temp, $collectivite_elem, $sql);
    }
}
```

Vous avez de nombreux exemples d'utilisation d'état et de sous-état dans les applications openMairie.

Une utilisation originale a été faite pour le Cerfa du recensement dans openRecensement où à la place du logo il a été mis une image du Cerfa.

On ne peut cependant pas faire tous les états et il est fort possible que vous ayez des états spécifiques. Vous avez des

exemples d'utilisation spécifique des méthodes de fpdf dans openElec : carte électorale, liste électorale,...

Vous pouvez compléter votre information avec le chapitre *framework/edition* et regarder les possibilités de paramétrage du générateur *generateur/parametrage* pour la réalisation d'état customisé.

Vous avez maintenant terminé l'exemple d'utilisation du Framework, le chapitre suivant a pour but de vous informer sur l'usage du framework..

3.1 Ergonomie

3.1.1 Ergonomie générale

L'application, sur la grande majorité des écrans, conserve ses composants disposés exactement au même endroit. Nous allons décrire ici le fonctionnement et l'objectif de chacun de ces composants. Cette structuration de l'application permet donc à l'utilisateur de toujours trouver les outils au même endroit et de se repérer rapidement.

Note : Les actions et affichages de l'application diffèrent en fonction du profil de l'utilisateur. Il se peut donc que dans les paragraphes qui suivent des actions soient décrites et n'apparaissent pas sur votre interface ou inversement que des actions ne soient pas décrites mais apparaissent sur votre interface.

3.1.1.1 Le logo

C'est le logo de l'application, il vous permet en un seul clic de revenir rapidement au tableau de bord.

3.1.1.2 Les actions personnelles

Cet élément affiche plusieurs informations importantes.

La première information est l'identifiant de l'utilisateur actuellement connecté ce qui permet de savoir à tout moment si nous sommes bien connectés et avec quel utilisateur. Ensuite est noté le nom de la collectivité sur laquelle nous sommes en train de travailler. En mode multi, une action est disponible sur cette information pour permettre de changer de collectivité. Ensuite la liste sur laquelle nous sommes en train de travailler, une action est disponible sur cette information pour permettre de changer de liste. Enfin l'action pour permettre de changer de mot de passe et pour se déconnecter sont disponibles en permanence.

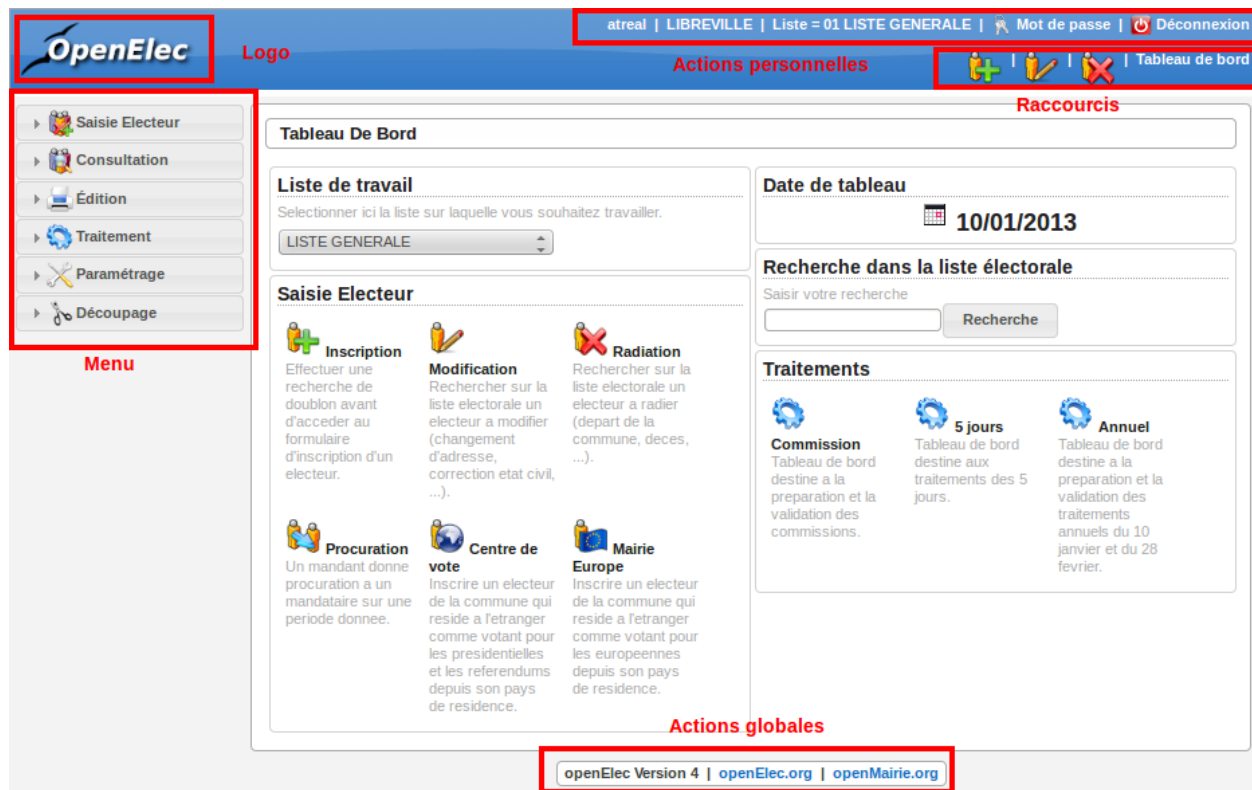


Fig. 1 – Ergonomie générale

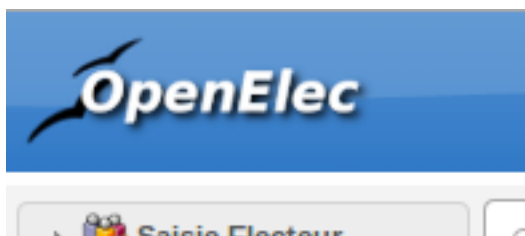


Fig. 2 – Logo

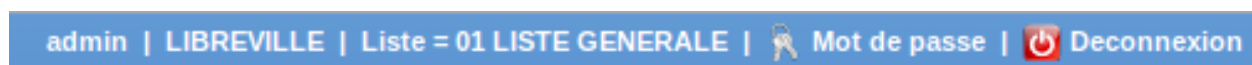


Fig. 3 – Actions personnelles

3.1.1.3 Les raccourcis

Cet élément permet d'afficher des raccourcis vers des écrans auxquels nous avons besoin d'accéder très souvent. Par exemple, ici nous avons des raccourcis directs vers les formulaires d'inscription, de modification et de radiation d'un électeur ainsi qu'un lien vers le tableau de bord.



Fig. 4 – Raccourcis

3.1.1.4 Le menu

Cet élément permet de classer les différents écrans de l'application en rubriques. En cliquant sur l'entête de rubrique, nous accédons à la liste des écrans auxquels nous avons accès dans cette rubrique.

Le nombre de rubriques disponibles dans le menu peut varier en fonction du profil des utilisateurs. Un utilisateur ayant le profil Consultation n'aura probablement pas accès aux six rubriques présentes sur cette capture.

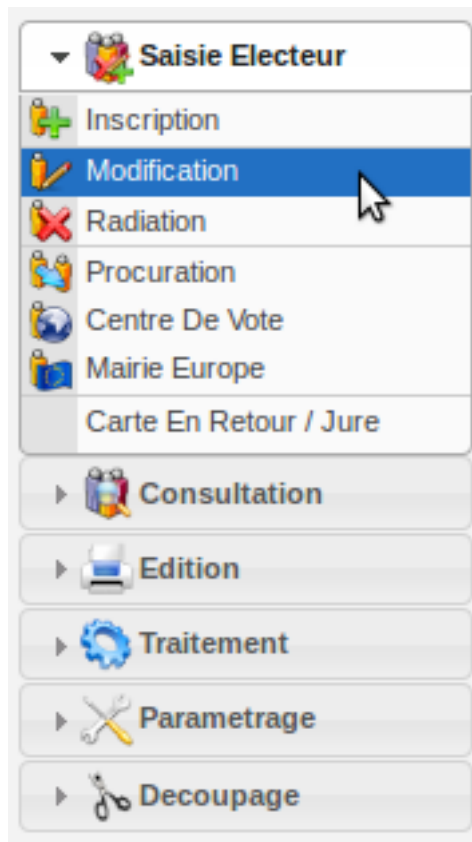


Fig. 5 – Menu

3.1.1.5 Les actions globales

Cet élément permet d'afficher en permanence le numéro de version du logiciel. Ensuite les différentes actions sont des liens vers le site officiel du logiciel ou vers la documentation.

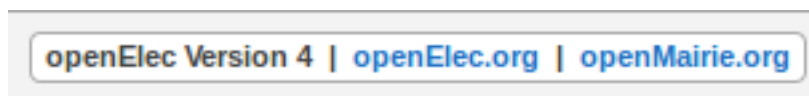


Fig. 6 – Actions globales

3.1.2 Usage du module om_sig

Ce document a pour objet de décrire le module sig interne d'openMairie dans la version om 4.4.5.

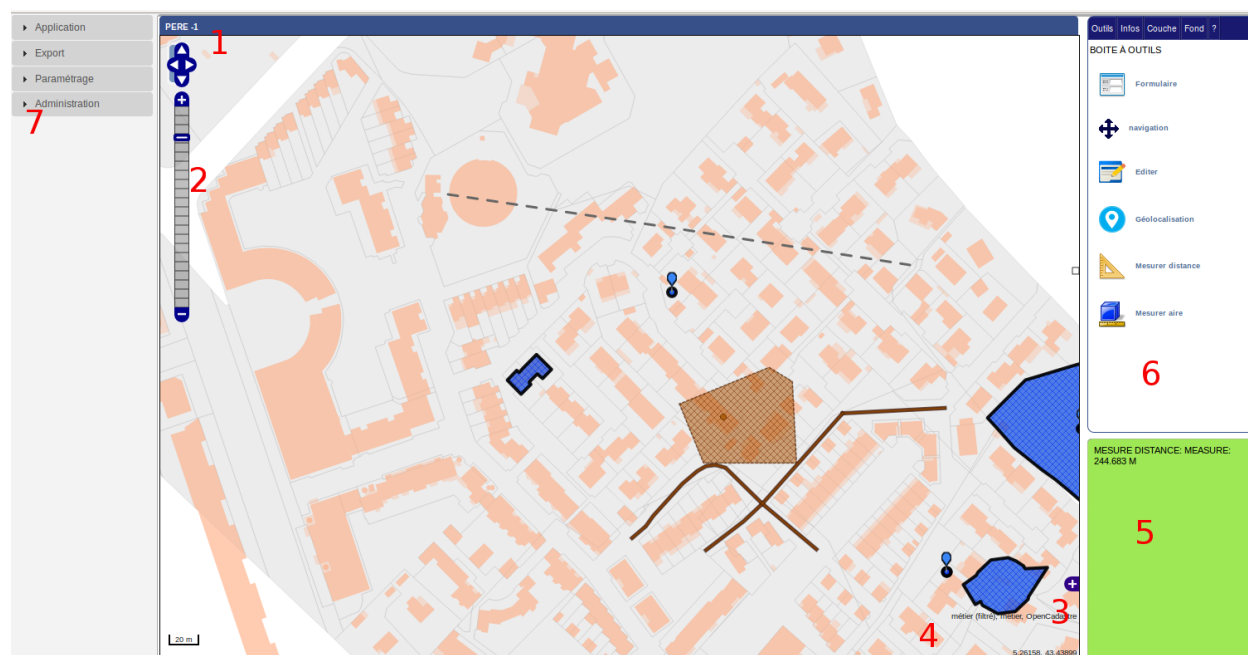
Dans sa version 4.4.5 ;

- intégration des formulaires dans le sig interne
- integration des résultats du moteur de recherche dans les cartes (cas utilisation moteur de recherche)
- intégration dans les cartes d'un résultat dans reqmo (cas d'utilisation reqmo)
- accès multiples aux objets
- accès à des objets multi géométrie

La nouveauté est la mise en place d'une nouvelle ergonomie avec un cartouche où sont accessibles toutes les commandes.

3.1.2.1 Ergonomie de l'interface SIG interne :

Nous allons décrire l'ergonomie d'om_sig qui se présente en plusieurs zones



Les éléments de l'interface sont les suivants :

1 En haut à gauche , il est noté l'objet et enregistrement concernés : objet PERE et enregistrement -1

Quand l'enregistrement est -1, cela veut dire qu'il n'y a pas d'objet sélectionné.

2 la barre de zoom

3 la fenêtre de navigation rapide

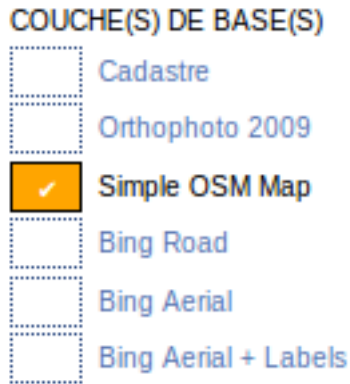
4 l'attribution : openCadastr

5 fenetre message : mesure distance = 244 683 m

6 Outils disponibles : boîte à outil, édition, information, couches et fonds

7 Menu

3.1.2.2 les fonds :



Ils sont paramétrés dans `om_sig_map`.

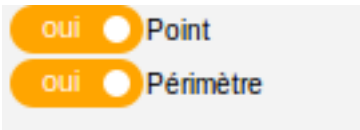
Dans notre cas, les options OSM et Bing sont cochés et il y a 2 flux wms paramétrés dans `om_sig_flux` et associé à `om_sig_map` (`om_sig_map_flux`) qui sont : cadastre, orthophoto 2003.

voir paramétrage

3.1.2.3 les couches :

Dans notre exemple, il y a :

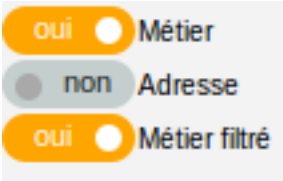
— deux couches vecteurs modifiables : point et périmètre,



— une couche de marqueurs (option `layerInfo` d'`om_sig_map`),



— Trois flux wms : métier, adresse et métier filtré



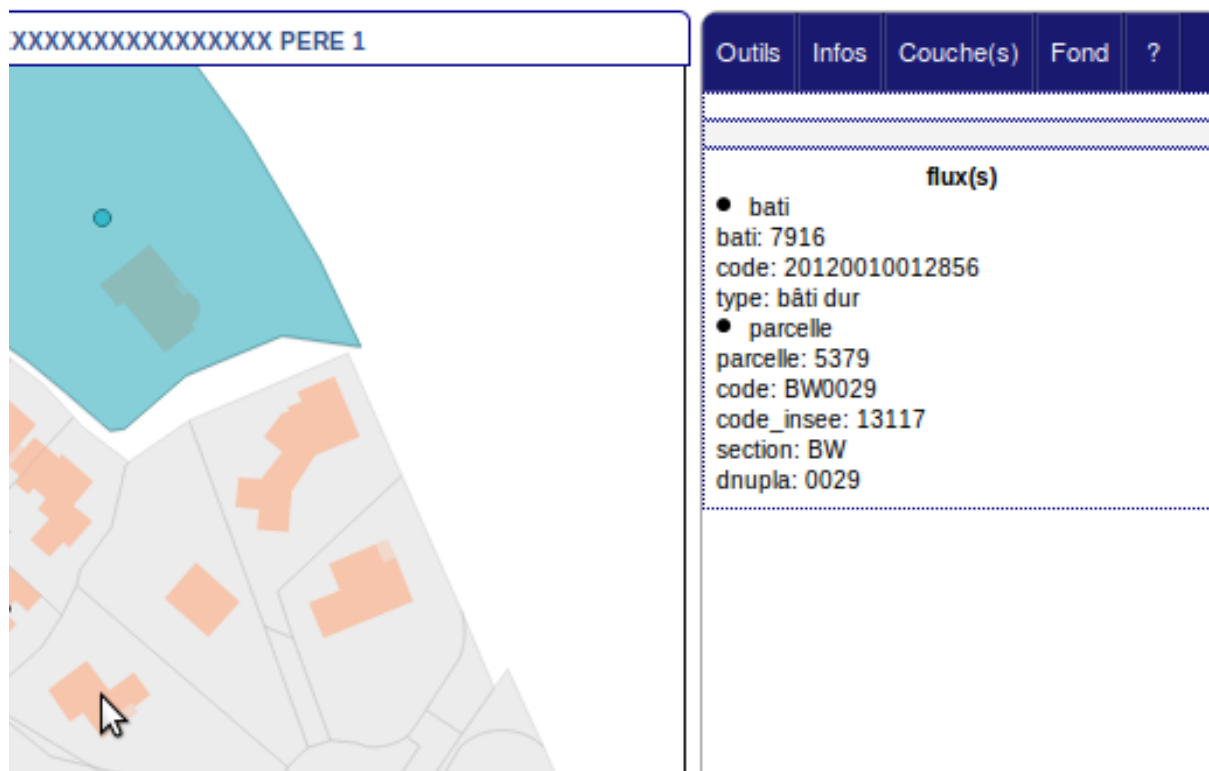
Les flux sont paramétrées dans `om_sig_flux` et elles sont associées aux cartes dans `om_sig_map_flux`.

Voir paramétrage

3.1.2.4 Information :

Cet onglet donne les informations disponibles lorsque l'on clique sur la carte sur un fond wms, le ou les marqueurs, une donnée vecteur.

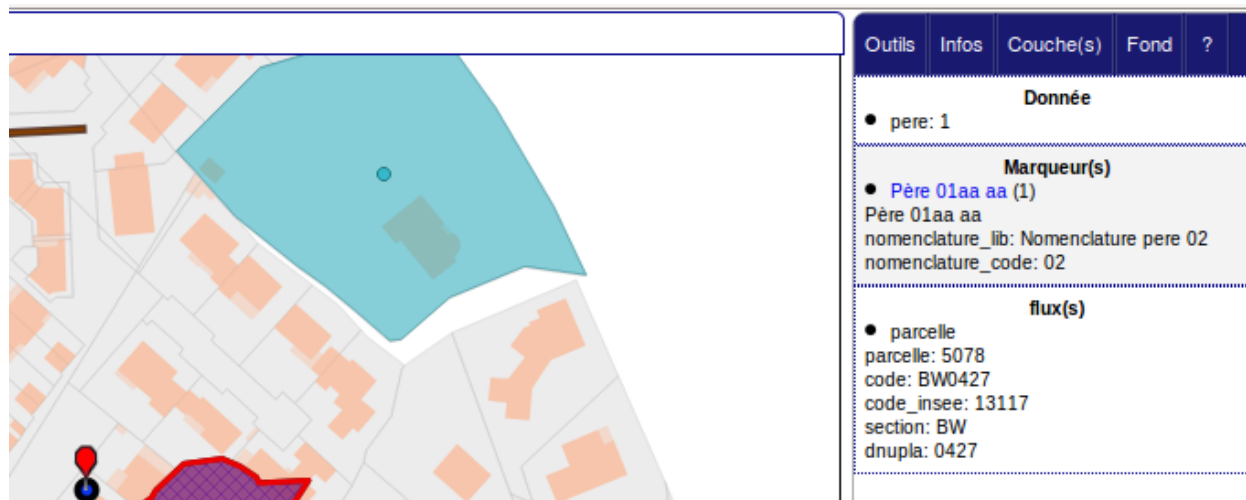
3.1.2.4.1 un fond wms



La couche de fond est le cadastre.

En cliquant sur le bâtiment 7916, le flux wms « bati » et le flux « parcelles » sont affichés dans l'onglet « infos »

3.1.2.4.2 le marqueur



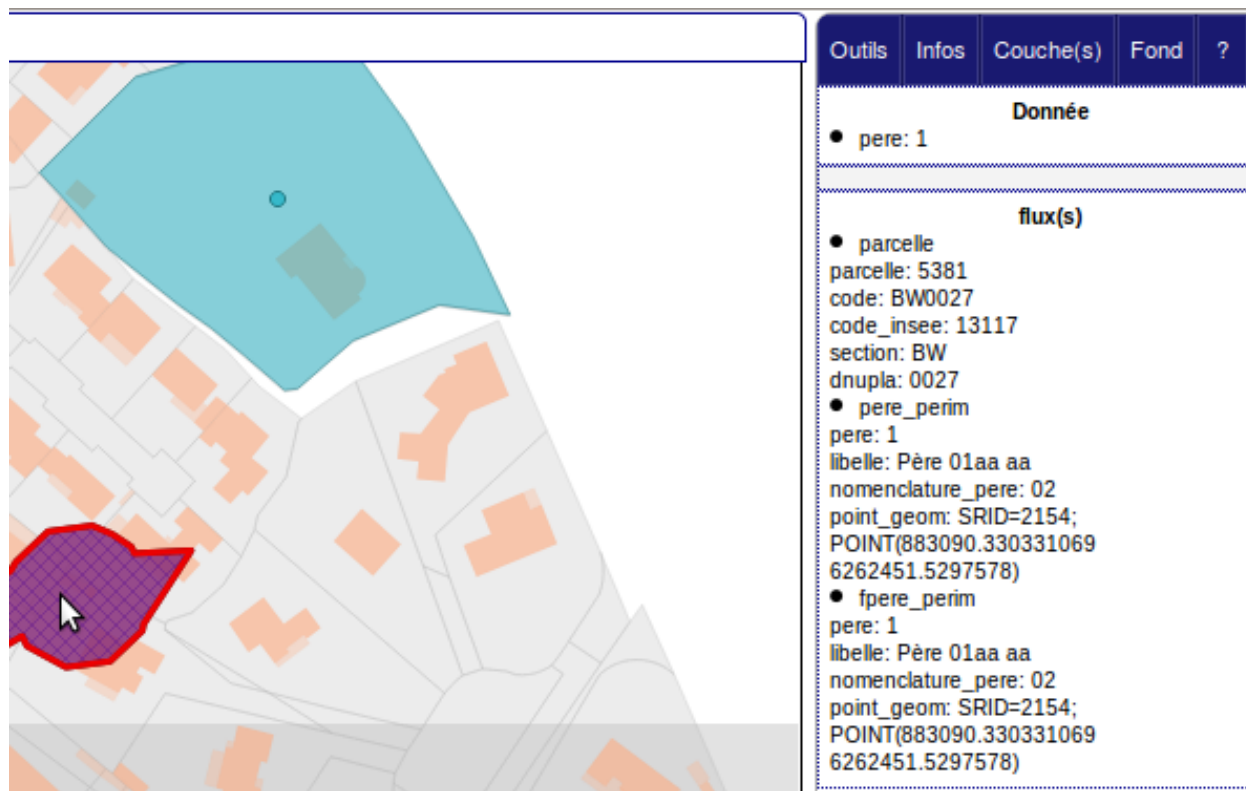
Lorsqu'il y a plusieurs enregistrements sur un même marqueur, exemple : plusieurs électeurs à une même adresse, tous les enregistrements s'affichent si on est sur une recherche simple ou sur un moteur de recherche.

Il est possible dans l'information du marqueur de mettre un pointeur vers un formulaire de la manière suivante :

Dans om_sig_map : champ URL

```
../scr/sousform.php?obj=pere&action=3&idx=
```

3.1.2.4.3 une donnée vecteur



The screenshot displays the openMairie Framework interface. On the left, a map shows a street layout with orange buildings and a large blue area. A purple polygon with a red border is highlighted on the map, and a mouse cursor is pointing at it. On the right, a panel titled 'Donnée' (Data) is visible. It has tabs for 'Outils', 'Infos', 'Couche(s)', 'Fond', and '?'. The 'Infos' tab is selected, showing details for a selected feature. The data is organized into sections: 'pere: 1' and 'flux(s)'. The 'flux(s)' section contains two entries, each with a list of attributes.

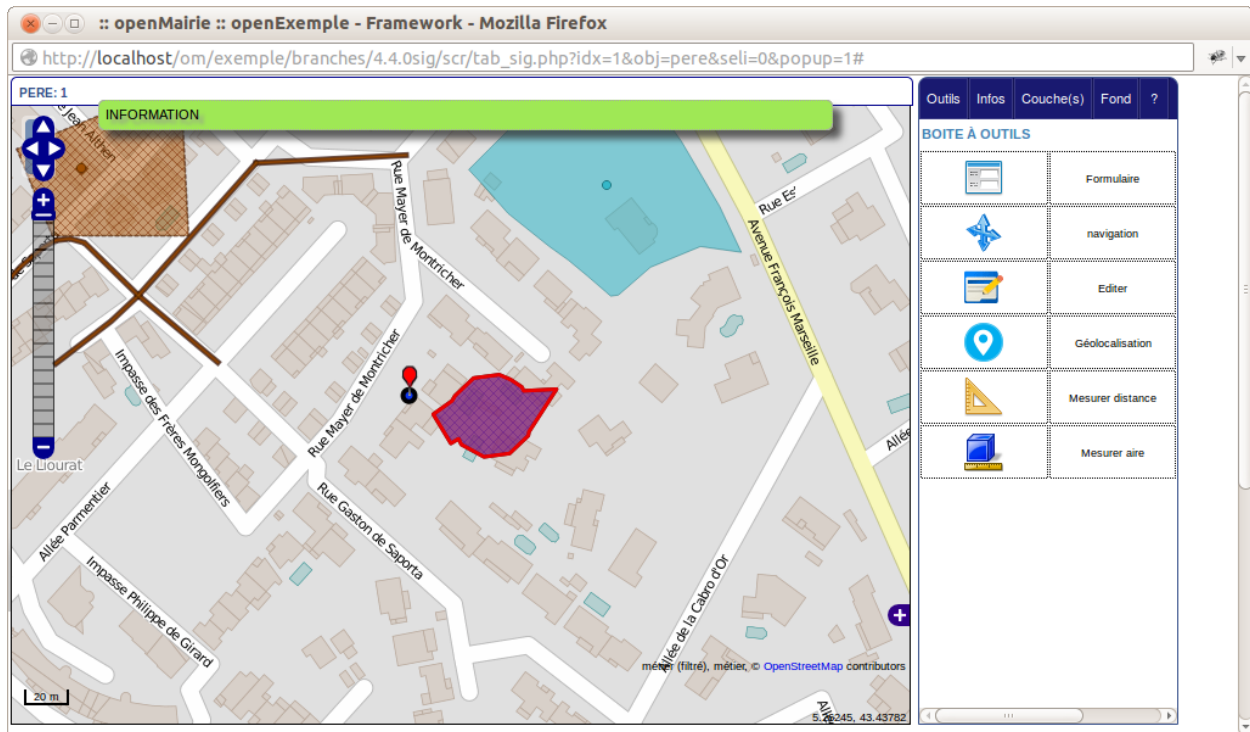
Donnée	
● pere: 1	
flux(s)	
● parcelle	
parcelle: 5381	
code: BW0027	
code_insee: 13117	
section: BW	
dnupla: 0027	
● pere_perim	
pere: 1	
libelle: Père 01aa aa	
nomenclature_pere: 02	
point_geom: SRID=2154;	
POINT(883090.330331069	
6262451.5297578)	
● fpere_perim	
pere: 1	
libelle: Père 01aa aa	
nomenclature_pere: 02	
point_geom: SRID=2154;	
POINT(883090.330331069	
6262451.5297578)	

les données sont paramétrables dans om_sig_map (voir paramétrage)

Les flux parcelles, pere_perim et fpere_perim sont les informations des flux wms cochés dans couche(s)

3.1.2.5 Boîte à outils :

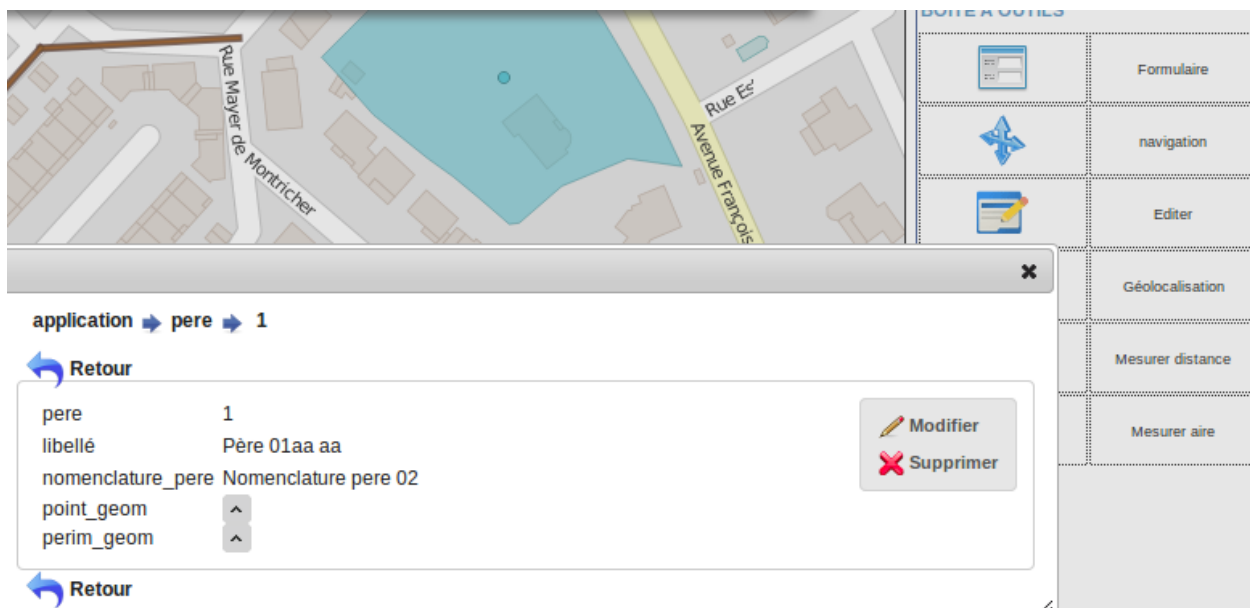
La boîte à outil est accessible dans l'onglet outil du menu cartographique



3.1.2.5.1 Accès au formulaire de saisie de données



Il est possible d'accéder au formulaire de saisie de l'enregistrement courant (sous formulaire)



En appuyant sur modifier, vous pouvez modifier les données de l'enregistrement.



La suppression de l'enregistrement n'est pas gérée dans la carte, par contre il est effectif dans la base :

- le point reste sur la carte et n'est pas supprimé
- les données du marqueur restent visualisables

En cas de rafraîchissement de la carte, les données ont disparues.

CONSEIL : ne pas utiliser l'option supprimer dans le formulaire

De même le champ geom du formulaire renvoie par défaut sur la carte. Il vaut mieux éviter d'afficher le champ geom sur les formulaires de cartes.

3.1.2.5.2 Navigation



Ce bouton sert à sortir des options de mesure et à revenir à la navigation.

3.1.2.5.3 Se géolocaliser dans la carte

Ce bouton sert à se géolocaliser dans la carte



3.1.2.5.4 Mesurer une distance

Il est possible de mesurer une distance avec l'outil



Cliquer sur les points à mesurer, cliquer 2 fois pour obtenir la mesure qui s'affiche dans la fenêtre d'observation

Appuyer sur le bouton « Navigation » pour sortir de l'outil de mesure.

3.1.2.5.5 Mesurer une aire

Il est possible de mesurer une aire avec l'outil



cliquer sur les angles du polygone à mesurer, cliquez 2 fois pour obtenir la mesure qui s'affiche dans la fenêtre observation

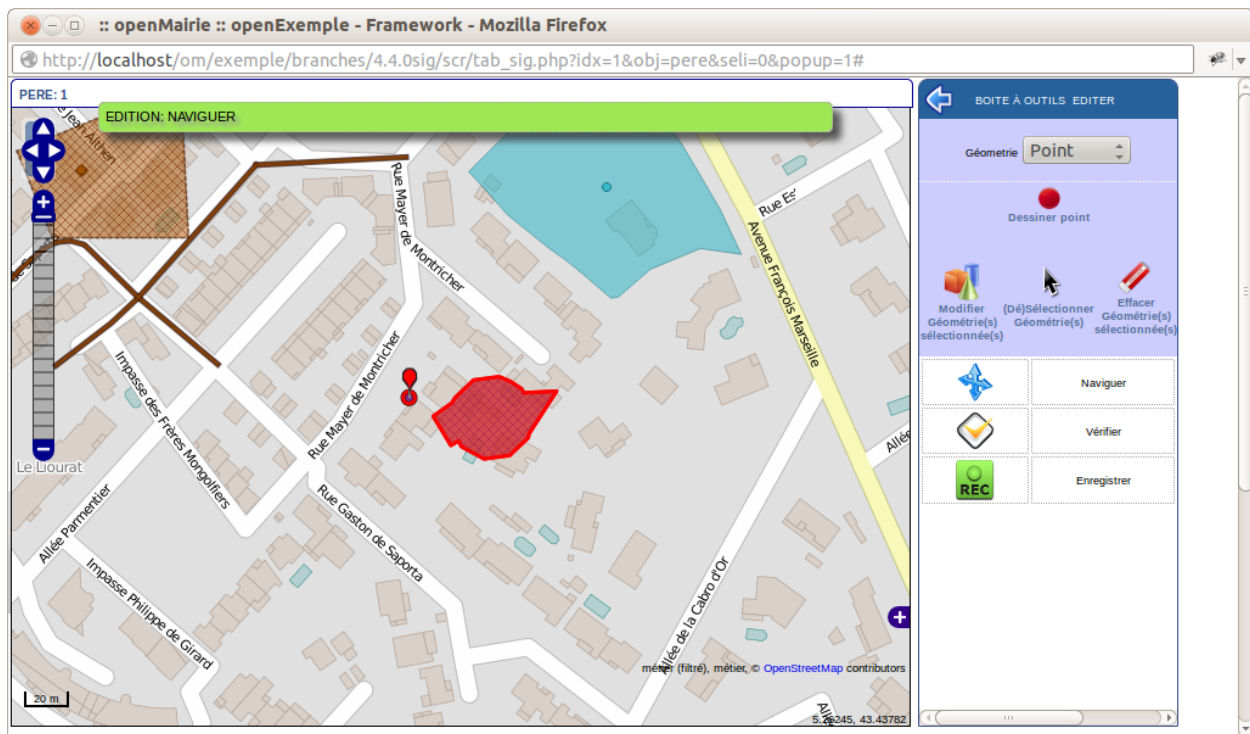
Appuyer sur le bouton « Navigation » pour sortir de l'outil de mesure.

3.1.2.6 Mode édition :

En mode édition, on ne peut plus accéder aux autres onglets

Cet onglet permet de modifier la ou les géométries de l'enregistrement de l'objet courant :

ci dessous la géométrie point de pere 1



Il est possible dans la fenêtre du haut de choisir une des géométries à modifier, ici le point ou le polygone de père 1

3.1.2.7 Edition d'un point :

Nous choisissons d'éditer le point :

3.1.2.7.1 Création d'un ou plusieurs points :



Après avoir sélectionner ce bouton, cliquez sur la carte à l'endroit où vous voulez le point Vous pouvez créer un ou plusieurs points. Le point est de couleur bleu

3.1.2.7.2 Modifier une géométrie sélectionnées :



Sélectionner un des points en cliquant dessus, il devient rouge.

Vous pouvez maintenant le déplacer

3.1.2.7.3 (Dé)sélectionner une géométrie :

Vous pouvez sélectionner ou désélectionner un point :



- bleu : non sélectionné
- rouge sélectionné

Un point sélectionné est actif pour une modification, suppression ou enregistrement

3.1.2.7.4 Supprimer une géométrie sélectionner :

En appuyant sur



Vous effacer la ou les géométries sélectionnées

3.1.2.7.5 Vérifier avant enregistrement d'un point :



Cette option vous permet de vérifier que votre géométrie est valide avant enregistrement.

Si vous avez par exemple plusieurs points sélectionnés et que la géométrie attendu est un seul point, un message s'affichera en observation

Edition: Données invalides! Point: MultiPoint sélectionné, point attendu

Si dans le même cas vous avez sélectionné qu'un seul point :

- Les points construits non sélectionnés seront effacés.
- le message sera le suivant

Edition: vérification terminée avec succès

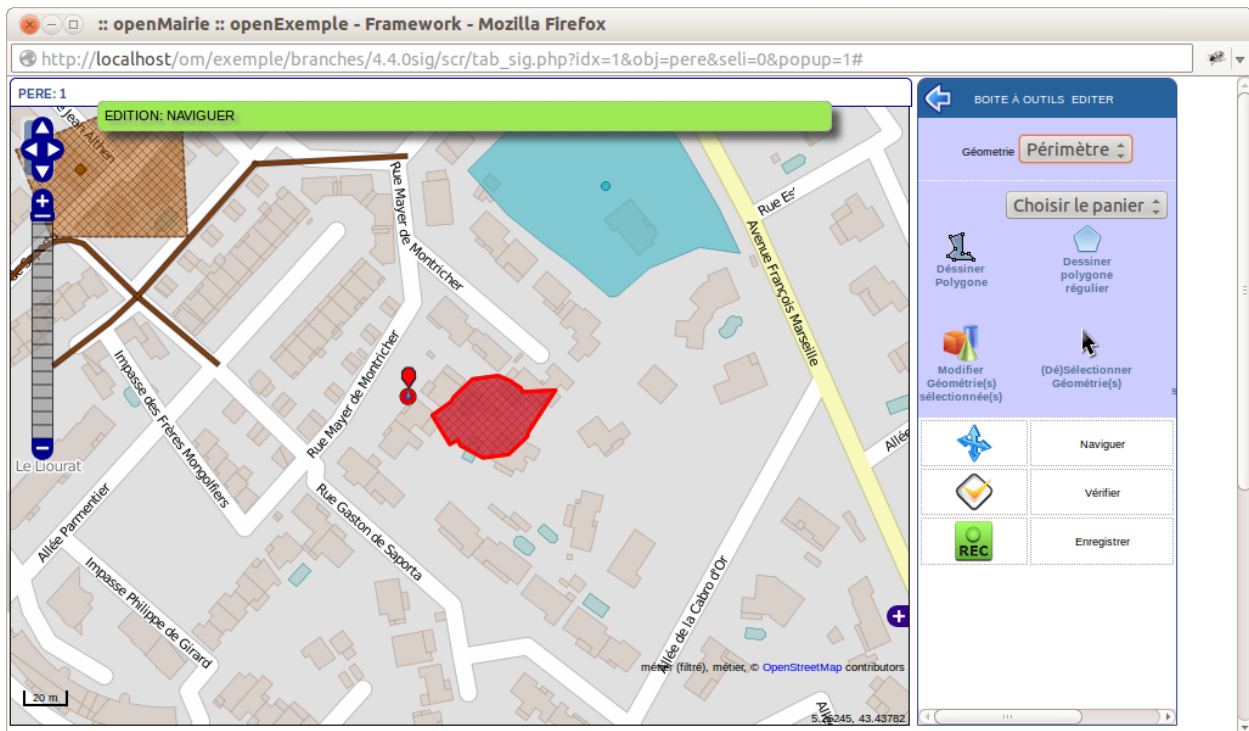
3.1.2.7.6 Enregistrer un point :

Cette option permet d'enregistrer un point.



3.1.2.8 Edition d'un polygone :

En sélectionnant périmètre, on peut mettre à jour la géométrie polygone.

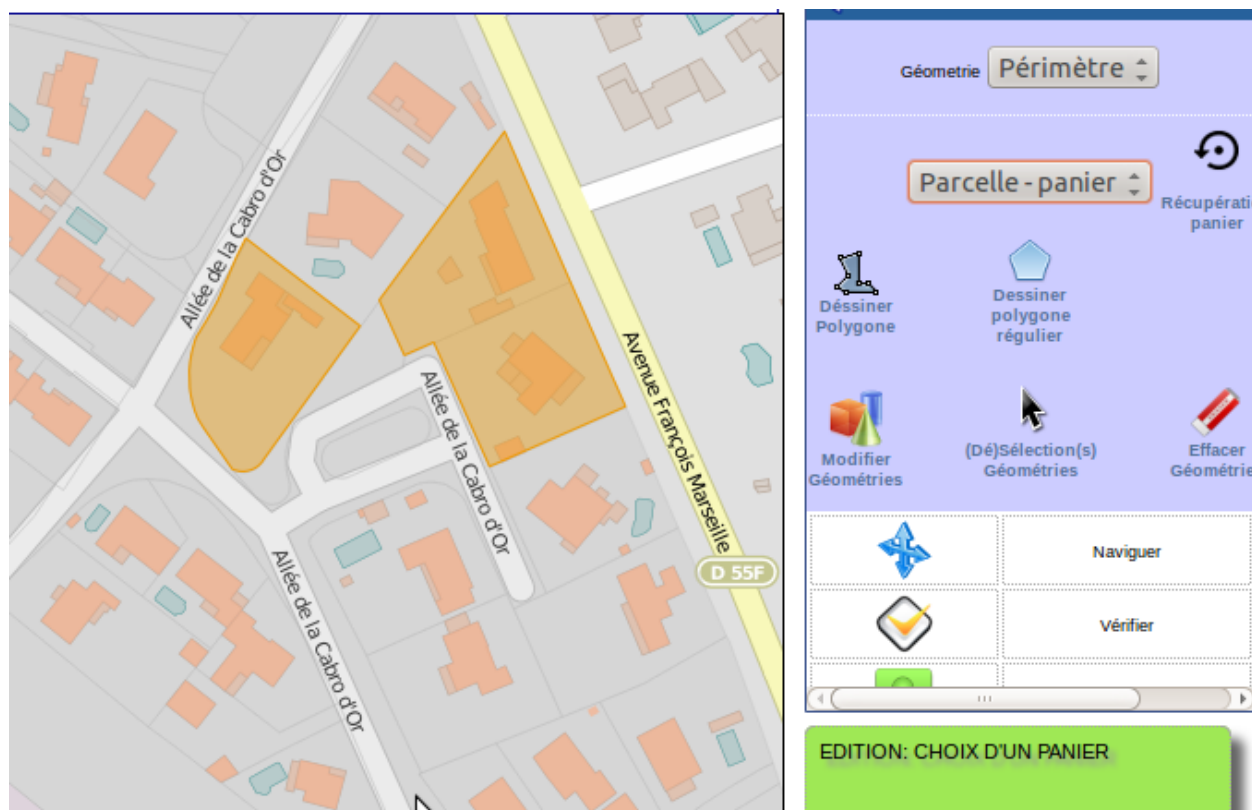


3.1.2.8.1 Utilisation du panier pour construire une géométrie

Dans panier, choisir un panier (ici parcelle panier)

Le fond correspondant à parcelle panier s'affiche (cadastre)

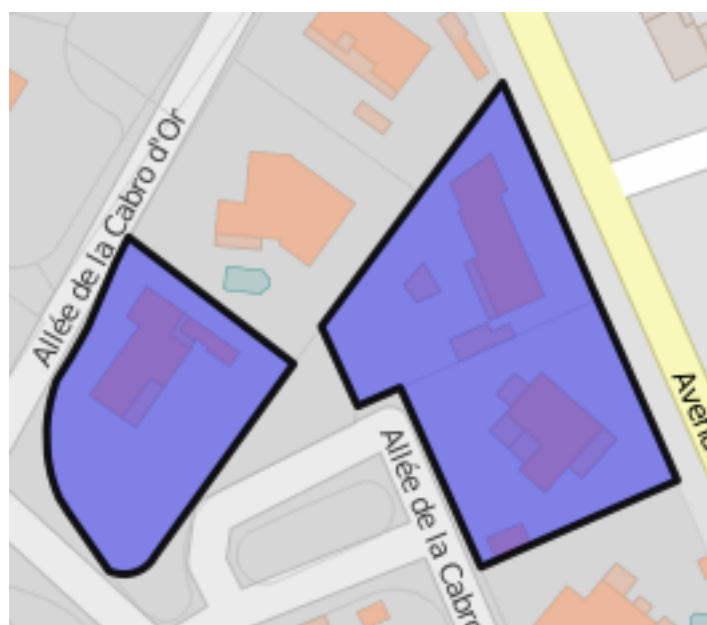
Sélectionner une ou des géométries.



Valider l'option récupération panier en appuyant sur



les objets récupérés sont en bleu.



3.1.2.8.2 Créer un polygone

Vous pouvez créer un polygone en appuyant sur :



Vous pouvez construire un polygone régulier en

— en appuyant sur



— sélectionner le nombre de côté que vous voulez (par défaut 4)

3.1.2.8.3 Modifier un polygone sélectionné :



Sélectionner un des polygones en cliquant dessus, il devient rouge.

Vous pouvez maintenant le modifier

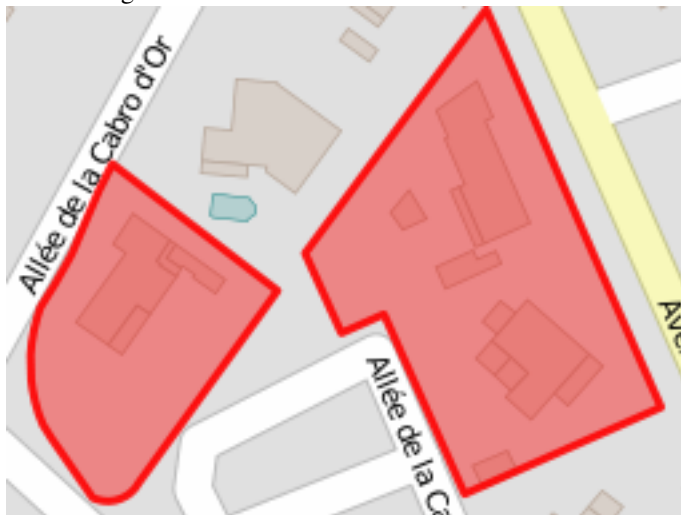
3.1.2.8.4 (Dé)sélectionner une géométrie :

Vous pouvez sélectionner ou désélectionner un polygone :



— bleu : non sélectionné

— rouge sélectionné



Un polygone sélectionné est actif pour une modification, suppression ou enregistrement

3.1.2.8.5 Supprimer un polygone sélectionné :

En appuyant sur



Vous effacer la ou les géométries sélectionnées

3.1.2.8.6 Vérifier avant enregistrement d'un polygone :



Cette option vous permet de vérifier que votre géométrie est valide avant enregistrement.

3.1.2.8.7 Enregistrer un polygone :

Cette option permet d'enregistrer un polygone.



3.1.2.9 Modification d'un ligne :

En sélectionnant ligne, on peut mettre à jour la géométrie ligne.

3.1.2.9.1 Utilisation du panier pour construire une géométrie

Dans panier, choisir un panier (ici tronçon panier)

Le fond correspondant à tronçon panier s'affiche

Sélectionner une ou des géométries.

Valider l'option récupération panier en appuyant sur



les objets récupérés sont en bleu.

3.1.2.9.2 Créer une ligne :

Vous pouvez créer une ligne en appuyant sur :



3.1.2.9.3 Modifier une ligne sélectionnée :



Sélectionner une des lignes en cliquant dessus, elle devient rouge.

Vous pouvez maintenant modifier les points de la ligne

3.1.2.9.4 (Dé)selectionner une géométrie :

Vous pouvez sélectionner ou désélectionner une ligne :



- bleu : non sélectionnée
- rouge sélectionnée

La ligne sélectionnée est active pour une modification, suppression ou enregistrement

3.1.2.9.5 Supprimer une ligne sélectionnée :

En appuyant sur



Vous effacer la ou les géométries sélectionnées

3.1.2.9.6 Vérifier avant enregistrement d'une ligne :



Cette option vous permet de vérifier que votre géométrie est valide avant enregistrement.

3.1.2.9.7 Enregistrer une ligne :



Cette option permet d'enregistrer une ligne.

3.2 Administration

Cette rubrique est dédiée à l'administration des fonctionnalités disponibles depuis l'interface : les tableaux de bord, les éditions, le module om_sig, ...

3.2.1 Les tableaux de bord

3.2.1.1 Widget

...

3.2.1.2 Composition

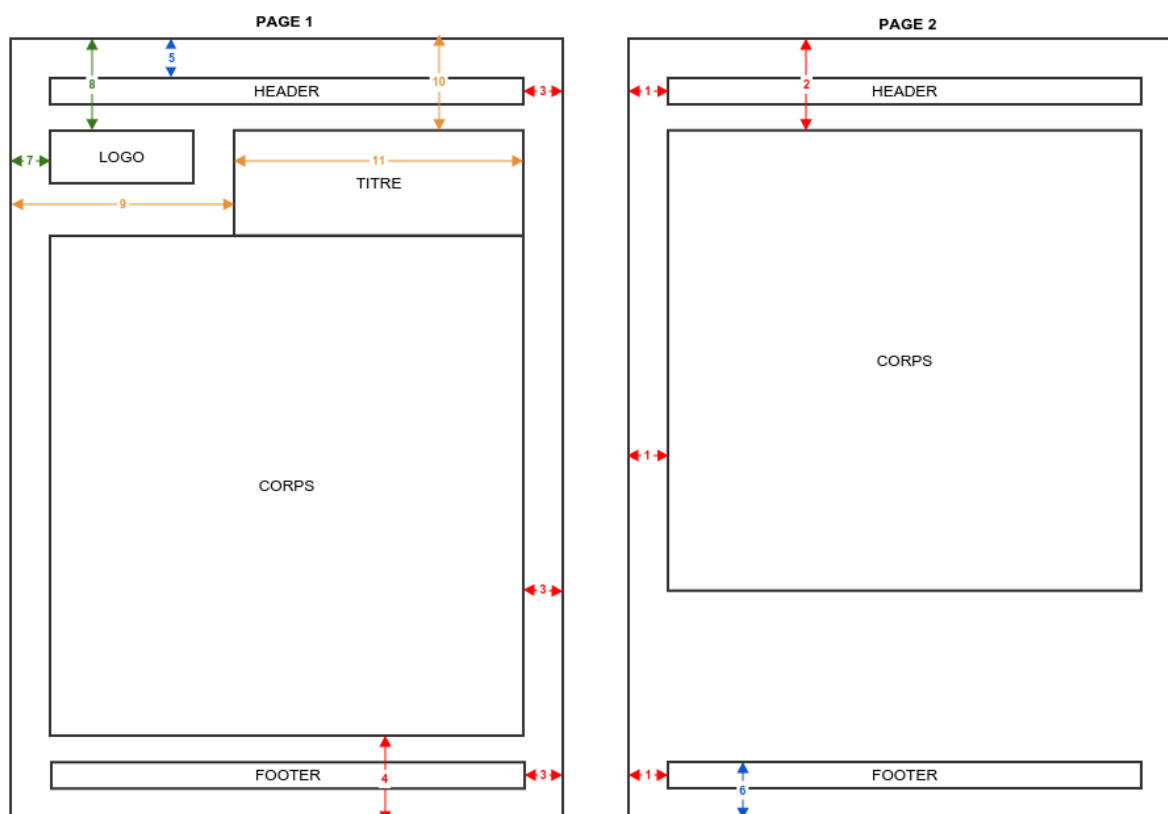
...

3.2.2 Les éditions

3.2.2.1 États et lettres types

Avertissement : Cette rubrique est en cours de rédaction.

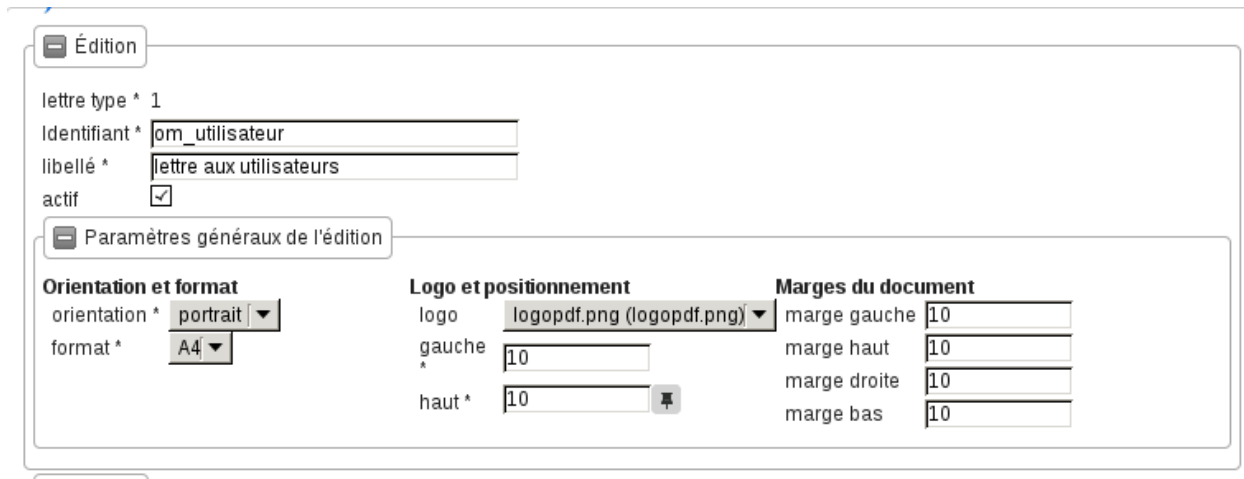
3.2.2.1.1 Positionnement des éléments dans l'édition



1. Marge gauche (en millimètre) de l'édition depuis la limite gauche de la page (voir *Bloc "édition"*) qui concerne les blocs : *Bloc "en-tête"*, *Bloc "corps"*, *Bloc "pied de page"*.
2. Marge haute (en millimètre) de l'édition depuis la limite haute de la page (voir *Bloc "édition"*) qui concerne le bloc : *Bloc "corps"*.

3. Marge droite (en millimètre) de l'édition depuis la limite droite de la page (voir *Bloc "édition"*) qui concerne les blocs : *Bloc "en-tête"*, *Bloc "corps"*, *Bloc "pied de page"*.
4. Marge basse (en millimètre) de l'édition depuis la limite basse de la page (voir *Bloc "édition"*) qui concerne le bloc : *Bloc "corps"*.
5. Espacement (en millimètre) entre le plafond du bloc "en-tête" et la limite haute de la page (voir *Bloc "en-tête"*).
6. Espacement (en millimètre) entre le plafond du bloc "pied de page" et la limite basse de la page (voir *Bloc "pied de page"*).
7. position du coin haut/gauche du logo par rapport au coin haut/gauche de l'édition (voir *Bloc "édition"*).
8. position du coin haut/gauche du logo par rapport au coin haut/gauche de l'édition (voir *Bloc "édition"*).
9. Espacement (en millimètre) entre la paroi gauche du bloc "titre" et la limite gauche de la page (voir *Bloc "titre"*).
10. Espacement (en millimètre) entre le plafond du bloc "titre" et la limite haute de la page (voir *Bloc "titre"*).
11. Largeur (en millimètre) du bloc "titre" depuis la paroi gauche du bloc "titre" (voir *Bloc "titre"*).

3.2.2.1.2 Bloc "édition"



Les informations d'**édition** à saisir sont :

- **id** : identifiant de l'état/lettre type.
- **libellé** : libellé affiché dans l'application lors de la sélection d'une édition.
- **actif** : permet de définir si l'édition est active ou non.

Note : Les champs **id** et **libellé** sont obligatoires, les **id** actif sont uniques.

3.2.2.1.2.1 Paramètres généraux de l'édition

Les champs de **paramètres généraux de l'édition** à saisir sont :

- **Orientation** : orientation de l'édition (portrait/paysage).
- **Format** : format de l'édition (A4/A3).
- **Logo** : sélection du logo depuis la table des logos configurés.
- **Logo haut** : position du coin haut/gauche du logo par rapport au coin haut/gauche de l'édition (voir 8 dans *Positionnement des éléments dans l'édition*).
- **Logo gauche** : position du coin haut/gauche du logo par rapport au coin haut/gauche de l'édition (voir 7 dans *Positionnement des éléments dans l'édition*).

- **Marge gauche** : Marge gauche (en millimètre) de l'édition depuis la limite gauche de la page (voir 1 dans *Positionnement des éléments dans l'édition*).
- **Marge haut** : Marge haute (en millimètre) de l'édition depuis la limite haute de la page (voir 2 dans *Positionnement des éléments dans l'édition*).
- **Marge droite** : Marge droite (en millimètre) de l'édition depuis la limite droite de la page (voir 3 dans *Positionnement des éléments dans l'édition*).
- **Marge bas** : Marge basse (en millimètre) de l'édition depuis la limite basse de la page (voir 4 dans *Positionnement des éléments dans l'édition*).

3.2.2.1.3 Bloc “en-tête”



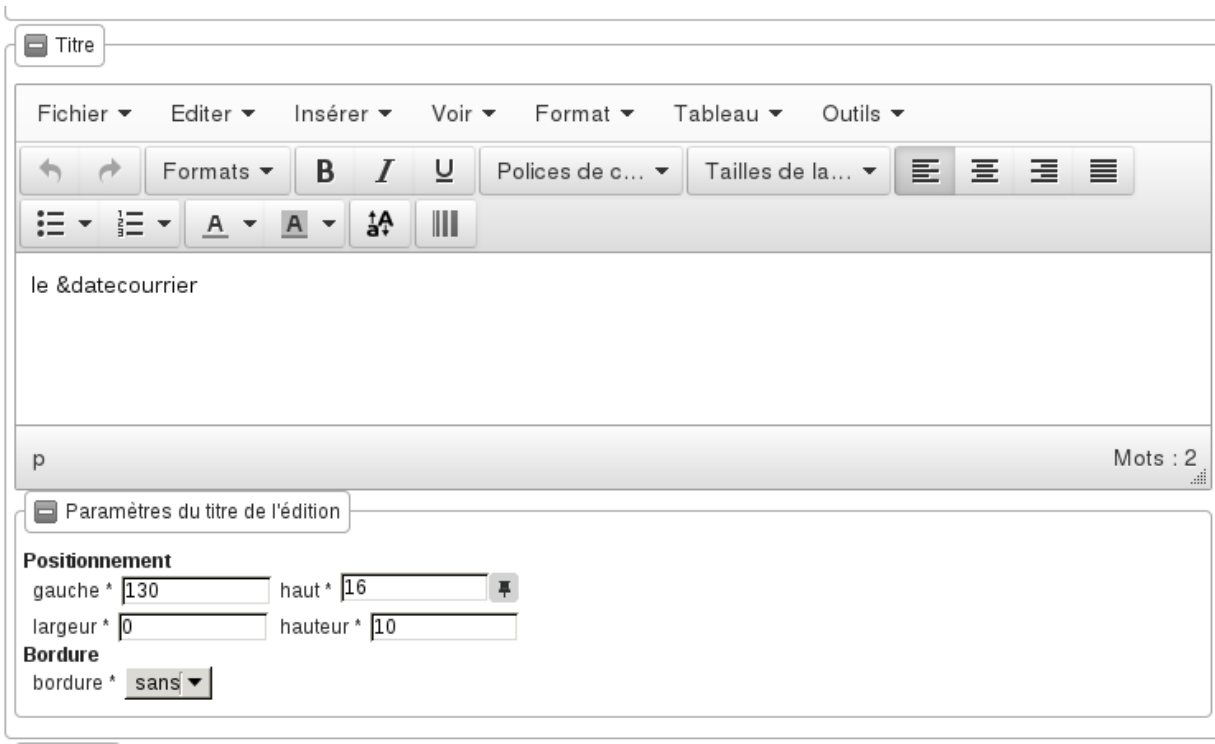
Ce bloc est facultatif et a pour particularité de se répéter sur chaque page.

- **Espacement** : Espacement (en millimètre) entre le plafond du bloc “en-tête” et la limite haute de la page (voir 5 dans *Positionnement des éléments dans l'édition*).
- **En-tête** : éditeur riche permettant une mise en page complexe.

Limitation(s) :

- les marges gauche et droite ne sont pas dissociables de celles du corps,
- la hauteur n'est pas fixable, elle dépend du contenu positionné à l'intérieur.

3.2.2.1.4 Bloc “titre”



Ce bloc est obligatoire.

- **Titre** : éditeur riche permettant une mise en page complexe.

3.2.2.1.4.1 Paramètres du titre de l'édition

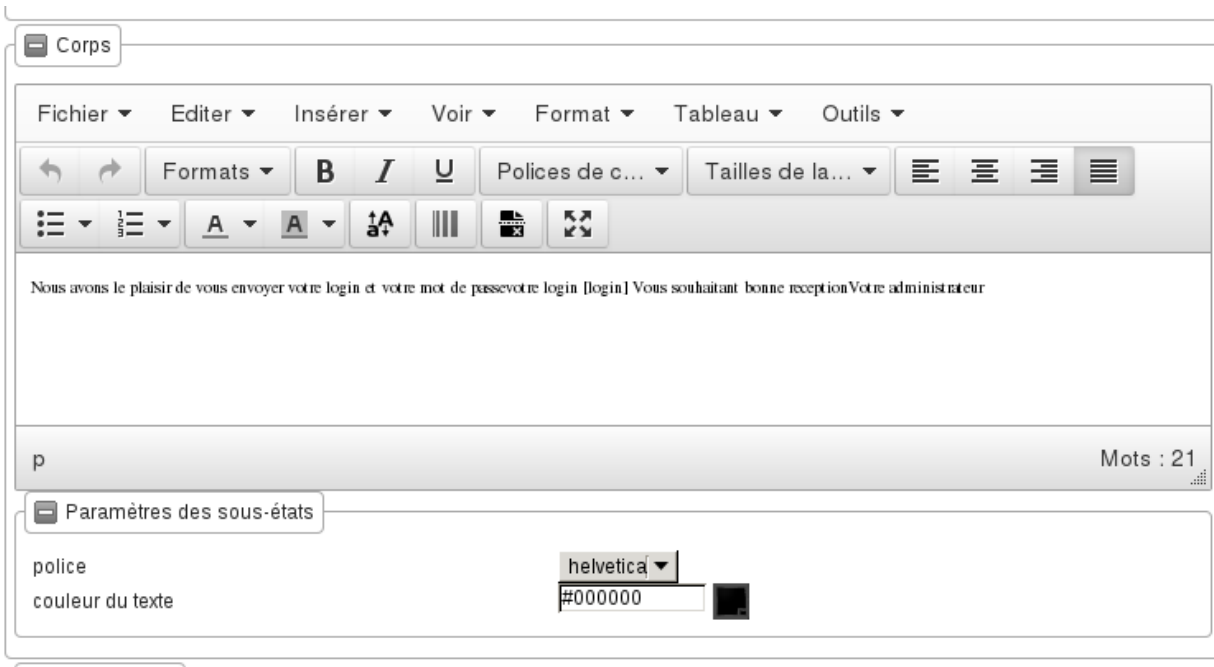
Positionnement :

- **Titre gauche** : Espacement (en millimètre) entre la paroi gauche du bloc “titre” et la limite gauche de la page (voir 9 dans *Positionnement des éléments dans l'édition*).
- **Titre haut** : Espacement (en millimètre) entre le plafond du bloc “titre” et la limite haute de la page (voir 10 dans *Positionnement des éléments dans l'édition*).
- **Largeur de titre** : Largeur (en millimètre) du bloc “titre” depuis la paroi gauche du bloc “titre” (voir 11 dans *Positionnement des éléments dans l'édition*).
- **Hauteur** : hauteur minimum du titre.

Bordure :

- **bordure** : Affichage ou non d'une bordure.

3.2.2.1.5 Bloc “corps”



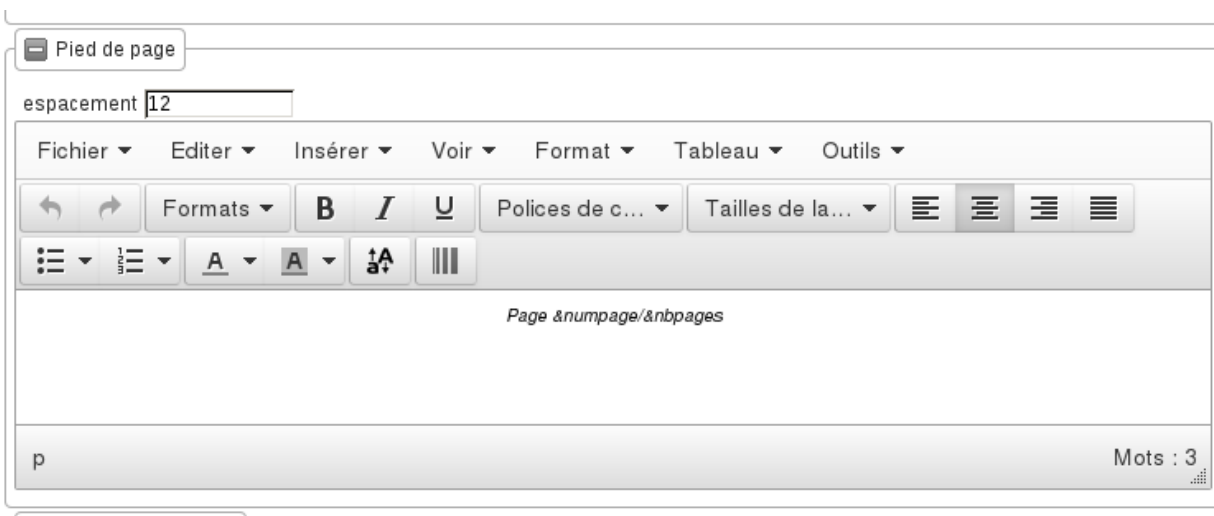
Ce bloc est obligatoire.

- **Corps** : éditeur riche permettant une mise en page complexe.

3.2.2.1.5.1 Paramètres des sous-états

- **Police personnalisée** : sélection de la police des sous-états.
- **Couleur texte** : sélection de la couleur du texte des sous-états.

3.2.2.1.6 Bloc “pied de page”



Ce bloc est facultatif et a pour particularité de se répéter sur chaque page.

- **Espacement** : Espacement (en millimètre) entre le plafond du bloc “pied de page” et la limite basse de la page (voir 6 dans *Positionnement des éléments dans l’édition*).
- **Pied de page** : éditeur riche permettant une mise en page complexe.

Limitation(s) :

- les marges gauche et droite ne sont pas dissociables de celles du corps,
- la hauteur n’est pas fixable, elle dépend du contenu positionné à l’intérieur.

3.2.2.1.7 Bloc “champs de fusion”



Champ(s) de fusion

requête * Requête SQL ▼

champ(s) de fusion

divers

&numpage Numéro de la page

&nbpages Nombre total de pages

variables de remplacement

- **Requête** : sélection d’un jeu de champs de fusion.
- **Champs de fusion** : Liste des champs de fusion disponibles pour la requête sélectionnée.
- **Variables de remplacement** : Liste des variables de remplacements disponibles.

3.2.2.1.8 Aide à la saisie dans les éditeurs de texte riche

3.2.2.1.8.1 Configurations disponibles

Trois configurations différentes de l’éditeur de texte riche sont utilisées :

- configuration n°1 : *Bloc “corps”*,
- configuration n°2 : *Bloc “en-tête”*, *Bloc “titre”*, *Bloc “pied de page”*,
- configuration n°3 : blocs de texte avec une mise en forme limitée toujours destinés à être intégré dans une édition via un champs de fusion.

Fonction / Configuration	N°1	N°2	N°3
<i>Gestion des tableaux</i>	x	x	
<i>Tableaux insécables</i>	x		
<i>Gestion des sauts de page</i>	x		
<i>Gestion des code-barres</i>	x	x	
<i>Majuscule/Minuscule</i>	x	x	x
<i>Insertion de sous-états</i>	x		
<i>Gestion du mode plein écran</i>	x	x	
<i>Code source</i>	x	x	x
<i>Correction orthographique</i>	x	x	x

3.2.2.1.8.2 Gestion des tableaux

Cette aide à la saisie n’est pas nécessairement disponible dans toutes les configurations de l’éditeur de texte riche (voir *Configurations disponibles*).

- **Créer un tableau** :

Choisir le nombre de lignes et de colonnes du tableau.

Note : Il faut bien placer le curseur dans une des cellules du tableau que l'on souhaite paramétrer. Idem pour le paramétrage des lignes et colonnes.

— **Paramétrage général du tableau :**

— Largeur :

Ce champ sert à indiquer la largeur du tableau en % (UNIQUEMENT) par rapport à la largeur du PDF. Par exemple, si le PDF fait une largeur de 30 cm et que la largeur du tableau est de 10%, le tableau fera 3 cm de largeur sur le PDF.

— Hauteur :

Ce champ sert à indiquer la hauteur du tableau en % (UNIQUEMENT) par rapport à la hauteur du PDF. Par exemple, si le PDF fait une hauteur de 50 cm et que la hauteur du tableau est de 25%, le tableau fera 12.5 cm de hauteur sur le PDF.

— Espacement inter-cellules :

Espacement entre les cellules. En pixel.

— Espace interne cellule :

Espacement entre les bords de la cellule et son contenu. En pixel.

— Bordure :

Epaisseur des bordures du tableau. En pixel.

— Titre :

Lorsque cette case est cochée, elle permet de rajouter un titre au tableau.

— Alignement :

Permet de choisir le type d'alignement du texte dans le tableau. Valeurs possibles : n/a (aucun), Gauche, Centré, Droite.

— **Supprimer un tableau**

— **Paramétrage des cellules :**

— Largeur :

Ce champ sert à indiquer la largeur de la colonne en % (UNIQUEMENT) par rapport à la largeur du tableau.

Par exemple, si le tableau fait une largeur de 30 cm et que la largeur de la colonne est de 10%, la colonne fera 3 cm de largeur.

— Hauteur :

Ce champ sert à indiquer la hauteur de la colonne en % (UNIQUEMENT) par rapport à la hauteur du tableau.

Par exemple, si le tableau fait une hauteur de 50 cm et que la hauteur de la colonne est de 25%, la colonne fera 12.5 cm de hauteur.

— Type de cellule :

Permet de définir si c'est une cellule « normale » ou une cellule qui va servir d'en-tête dans le tableau. Valeurs possibles : Cellule, Cellule d'en-tête.

— Étendue :

Paramètre sur quoi doivent s'appliquer les paramètres renseignés. Valeurs possibles : n/a (aucun), Ligne, Colonne, Groupe de lignes, Groupe de colonnes.

— Alignement :

Permet de choisir le type d'alignement du texte dans la cellule. Valeurs possibles : n/a (aucun), Gauche, Centré, Droite.

— **Fusionner des cellules :**

En sélectionnant les cellules à fusionner et en cliquant sur Tableau → Cellule → Fusionner les cellules les cellules seront fusionnées.

Si aucune cellule n'est sélectionnée, un menu apparaît :

- Colonnes :

Nombre de colonnes qui vont être fusionnées à partir de la cellule dans laquelle le curseur est positionné.

- Lignes :

Nombre de lignes qui vont être fusionnées à partir de la cellule dans laquelle le curseur est positionné.

- **Diviser les cellules :**

Divise la cellule dans laquelle le curseur est positionné si elle avait été fusionnée avant.

- **Paramétrage des lignes :**

- Type de ligne :

Permet de définir le type de la ligne. Valeurs possibles : En-tête, Corps, Pied.

- Alignement :

Permet de choisir le type d'alignement du texte dans la ligne. Valeurs possibles : n/a (aucun), Gauche, Centré, Droite.

- Hauteur :

Ce champ sert à indiquer la hauteur de la ligne en % (UNIQUEMENT) par rapport à la hauteur du tableau.

Par exemple, si le tableau fait une hauteur de 50 cm et que la hauteur de la ligne est de 25%, la ligne fera 12.5 cm de hauteur.

- **Insérer une ligne :**

Permet d'insérer une ligne avant ou après la ligne sur laquelle le curseur est positionné.

- **Effacer une ligne :**

Supprimer la ligne sur laquelle le curseur est positionné.

- **Couper une ligne :**

Coupe la ligne sur laquelle le curseur est positionné.

- **Copier une ligne :**

Copie la ligne sur laquelle le curseur est positionné.

- **Coller une ligne :**

Colle la ligne qui avait été copiée/coupée avant ou après la ligne sur laquelle le curseur est positionné.

- **Insérer une colonne :**

Insère une colonne avant ou après la colonne sur laquelle le curseur est positionné.

- **Effacer une colonne :**

Supprime la colonne sur laquelle le curseur est positionné.

3.2.2.1.8.3 Tableaux insécables

Cette aide à la saisie n'est pas nécessairement disponible dans toutes les configurations de l'éditeur de texte riche (voir *Configurations disponibles*).

...

3.2.2.1.8.4 Gestion des sauts de page

Le saut de page permet d'insérer un marqueur dans l'édition PDF, pour que le contenu qui suit soit positionné sur la page suivante.

Cette aide à la saisie n'est pas nécessairement disponible dans toutes les configurations de l'éditeur de texte riche (voir *Configurations disponibles*).

Pour ajouter un saut de page, il faut :

- positionner le curseur là où l'on souhaite l'insérer,
- cliquer sur l'élément "Saut de page" du menu déroulant de l'éditeur de texte riche "Insérer".

Dans l'éditeur apparaît un rectangle avec des bordures en pointillés.

3.2.2.1.8.5 Gestion des code-barres

Cette aide à la saisie n'est pas nécessairement disponible dans toutes les configurations de l'éditeur de texte riche (voir *Configurations disponibles*).

Saisir le champ de fusion

Sélectionner le champ de fusion

Cliquer sur le bouton de génération du code-barres puis valider le formulaire pour enregistrer les changements

3.2.2.1.8.6 Majuscule/Minuscule

Cette aide à la saisie n'est pas nécessairement disponible dans toutes les configurations de l'éditeur de texte riche (voir *Configurations disponibles*).

...

3.2.2.1.8.7 Gestion du mode plein écran

Cette aide à la saisie n'est pas nécessairement disponible dans toutes les configurations de l'éditeur de texte riche (voir *Configurations disponibles*).

...

3.2.2.1.8.8 Code source

Cette aide à la saisie n'est pas nécessairement disponible dans toutes les configurations de l'éditeur de texte riche (voir *Configurations disponibles*).

...

3.2.2.1.8.9 Correction orthographique

Cette aide à la saisie n'est pas nécessairement disponible dans toutes les configurations de l'éditeur de texte riche (voir *Configurations disponibles*).

...

3.2.2.1.8.10 Insertion de sous-états

Cette aide à la saisie n'est pas nécessairement disponible dans toutes les configurations de l'éditeur de texte riche (voir *Configurations disponibles*).

...

3.2.2.2 Sous-états

3.2.2.3 Requêtes

3.2.2.4 Logos

3.2.3 Paramétrage du module om_sig

Il est décrit ici le paramétrage du sig interne.

Il est décrit les principes ainsi que les différents formulaires du sig interne.

Ces formulaires sont accessibles dans le menu option administration. Il est nécessaire que l'API openLayers soit dans le framework (il y est de base) :

lib/openlayers

3.2.3.1 Principes

Il est proposé dans ce chapitre de décrire le module sig interne qui permet la géo localisation d'objet dans openMairie

Depuis la version 4.4.0, le sig interne est accessible dans le framework en mettant le paramètre option_localisation d'om_parametre à la valeur « sig_interne ».

L'objectif de sig interne est de permettre une saisie le plus souvent automatique de géométries. Cette saisie est stockée dans la base métier postgresql. Elle est affichée sur des fonds existants sur internet : google sat, openStreemap ou bing (pour l'instant). Elle peut être affichée sur un flux (wms ou tiles)

Il n'est donc pas nécessaire de disposer d'un SIG pour utiliser le sig interne ;

Le format de stockage des données pgsq est celui de l'OGC et il est accessible aux clients libres où propriétaires qui respectent ce format. Par exemple qgis (outil libre) peut accéder aux données de la base postgres « métier ».

3.2.3.1.1 Pré requis

- base postgres 9.x
- postgis > 2.x

3.2.3.1.2 Géo localisation automatique sur une adresse postale

L'enjeu est de limiter au maximum la géo localisation manuelle dès qu'il y a une possibilité de géo localisation automatique.

Elle se fait au travers de 4 programmes :

- adresse_postale.php : positionnement suivant le numero et rue
- adresse_postale_google.php : positionnement suivant le numero et rue avec google
- adresse_postale_bing.php : positionnement suivant le numero et rue avec bing
- adresse_postale_mapquest.php : positionnement suivant le numero et rue avec mapquest

La géolocalisation automatique peut se faire sur une base externe postgresql (eventuellement via une vue)

Le paramétrage se fait dans dyn/adresse_postale.inc.php.

3.2.3.1.3 Affichage de carte

L'affichage se fait avec openLayers dont le composant est de base dans le framework openMairie : lib/openLayers. (le composant est installé de manière à être optimisé avec une css openmairie)

La librairie proj4 inclus dans lib/openLayers permet de pouvoir utiliser les projections lambert sud et lambert 93.

La projection géographique et Mercator est de base dans openLayers

L'enjeu est donc de projeter les données stockées dans la base « métier » postgresql - postgis (les communes devant utiliser le lambert93) en mercator pour être lisible avec les cartes accessibles sur internet.

L'affichage des marqueurs est fait au travers d'une requête postgresql qui alimente un tableau json lu comme une couche openLayers.

La data à modifier est fourni par requete postgresql au format wkt à openLayers.

le sig interne permet

```
- l'affichage de/des fond(s)
- l'affichage de marqueurs (data)
- l'affichage du géométries qui peut être créé ou déplacé (couche wkt)
```

dans la version 4.2.0, il permet

```
- l'affichage de flux wms et wfs (getmap) et de récupérer les données (getfeature)
- la collation de géométrie dans un panier et son enregistrement en multi géométries
```

dans la version 4.4.5 il a été rajouté

```
- la boîte à outils : édition du formulaire, édition, outils de mesure, ↵
↪ géolocalisation
- l'outil de saisie a été amélioré
- les cartouche info, couche et fond
```

Dans cette dernière version, le module est intégré dans openMairie et utilise les formulaires de saisie, le moteur de recherche et les requêtes mémorisées lors de l'affichage.

3.2.3.1.4 L'intégration dans openMairie

Le module est intégré à openMairie :

Le module SIG est accessible depuis le formulaire d'affichage (tab), par le bouton suivant :



Il prend en compte la recherche simple ou avancée du formulaire d'affichage.

Le module SIG est accessible depuis le module de requête mémorisée.

Il est possible d'éditer les attributs d'un objet sélectionné et de les modifier via le formulaire.

3.2.3.1.5 Paramétrage de la carte

Le paramétrage général des cartes est modifiable dans dyn/var_sig.inc

```
// $cle_google = "xxxxxxxxxxxxxxxxxxxx";
// $http_google="http://maps.google.com/maps?file=api&v=2&key=";
// $http_bing='http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=6.2&
→mkt=en-us'; // 6.3c au lieu de 6.2
// $fichier_jsons="sig_json.php?obj=";
// $fichier_wkt="sig_wkt.php";
// *** zoom par couche : zoom standard permettant un passage de zoom a l autre
// $zoom_osm_maj=18;
// $zoom_osm=14;
// $zoom_sat_maj=8;
// $zoom_sat=4;
// $zoom_bing_maj=8;
// $zoom_bing=4;
// *** popup data contenuHTML
// $width_popup=200;
// $cadre_popup=1;
// $couleurcadre_popup="black";
// $fontsize_popup=12;
// $couleurtitre_popup="black";
// $weighttitre_popup="bold";
// $fond_popup="yellow";
// $opacity_popup="0.7";
// *** localisation maj ou consultation
// $img_maj="../img/punaise_sig.png";
// $img_maj_hover="../img/punaise_hover.png";
// $img_consult="../img/punaise_point.png";
// $img_consult_hover="../img/punaise_point_hover.png";
// $img_w=14;
// $img_h=32;
// $img_click="1.3"; // multiplication hauteur et largeur image cliquee
```

Toutes ces variables ne sont plus accessibles dans la version 4.5.0 (à vérifier)

Le paramétrage de la projection qui est proposé dans le formulaire de saisie `om_sig_map` se paramètre dans `var_sig.inc.php`.

Il est décrit ci dessous son paramétrage par défaut

```
$contenu_epsg[0] = array("", "EPSG:2154", "EPSG:27563");
$contenu_epsg[1] = array("choisir la projection", 'lambert93', 'lambertSud');
```

Il est à noter que les étendues ne sont plus dans `var_sig` dans la version 4.4.5 et qu'elles sont stockées dans la table `om_sig_extent`.

3.2.3.1.6 Autre point d'entrée

Il est créé un `om_map.class.php` dans `obj` pour pouvoir mettre les points d'entrée

Nous décrivons ici les anciens points d'entrées de `form_sig.php` et de `tab_sig.php`

Dans `dyn/form_sig_update.inc.php`, il est possible de paramétrer des post traitements de saisie de géométrie

Dans `dyn/form_sig_delete.inc.php`, il est possible de paramétrer des post traitements de suppression de géométrie

Dans `dyn/tab_sig_barre.inc.php`, il est possible de personnaliser la fenêtre

La barre permet de modifier des champs exemple `openAdresse`

voie 502  **Data** 

Voie n°502 - ALLEE AUGUSTE CHABAUD

cde int. 13117#06H	insee. 13117	rivoli. 13117#06H	Numéro.
Ref. registre	Type ALLEE (ALL)	Lib. primaire	Mot directeur AUGUSTE CHABAUD
Lieu-dit	Couleur couleur 10	Source Registre	Statut non renseigné
Action non renseigné	Etat avanc. Validé	Typ. num. sérielle	Observat°

 Calculer les noms de champs

Nom complet ALLEE AUGUSTE CHA Nom abrégé ALL AUGUSTE CHAB Nom carto all auguste chabaud

 **Modifier voie**

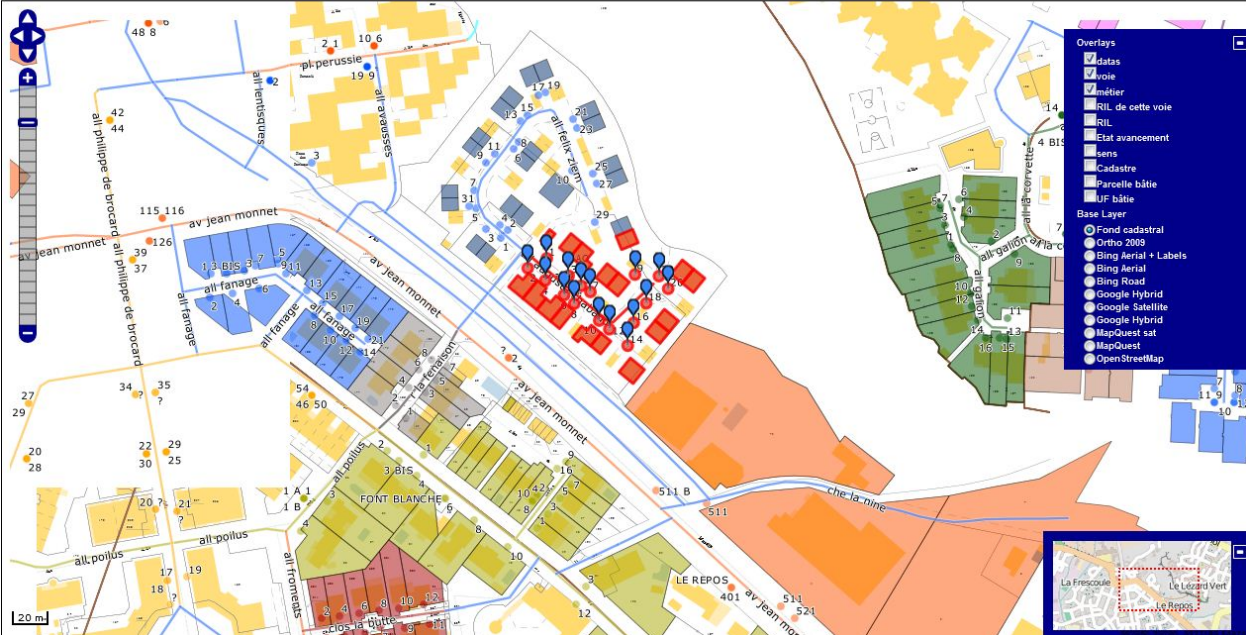
 Cal.empr.parcelle

 Cal.empr.dgi

 Mod.Adr.Lot

 Mod.Tronc.Lot

 Cal.nom.adresses



Overlays

- ☒ Datas
- ☒ Voie
- ☒ Métier
- ☒ RIL de cette voie
- ☒ RIL
- ☒ Etat avancement
- ☒ Sens
- ☒ Cadastre
- ☒ Parcelle bâtie
- ☒ UF bâtie

Base Layer

- ☒ Fond cadastral
- ☐ Ortho 2009
- ☐ Bing Aerial + Labels
- ☐ Bing Aerial
- ☐ Bing Road
- ☐ Google Hybrid
- ☐ Google Satellite
- ☐ Google Hybrid
- ☐ MapQuest sat
- ☐ MapQuest
- ☐ OpenStreetMap

La barre permet d'afficher les attributs d'un regard, et permet d'aller sur le point aval ou le point amont.

pluvial_regard 4214

4000 (4214)

Rue: bd de l'europe **Topo L93C44 X/Y/Z** : 1881636.98 / 3140424.42 / 67.846
Anomalie: RAS **Observations:**
Type: REGARD **Couverture:** TAMPON_FONTE **Nb piq.:** 0 **Diam.:** 0 **Larg.:** 0.6 **Long.:** 0.6 **Matériaux:** BETON **Etat:** BON
Radier: prof. 2.36, mat.: BETON, état: BON, fe: 65.486, **Echelon:** OUI, état: BON
Autres equip.: gaz. Non, AEP: Non, EU: Non, arr.: Non, FO: Non **Dysfonc.:** dépôt. Non, MES.: Non, écou.: Non, eau par.: NON, des.:
Dte saisie.: 2011-11-28, **météo:** SEC, **ech.:** 1/5000, **source:** IPSEAU_INGEROP, **topo:** Géomètre, **corr.:**
Amont 1: 4028 (4275) , type CIRCULAIRE, diam 400, l/h/p 0/0/2.36, mat BETON, fe 65.486
aval 1: 4001 (4215) , type CIRCULAIRE, diam 400, l/h/p 0/0/2.36, mat BETON, fe 65.486

Formulaire
 Fossés DWG
 Fossés PDF
 Photo

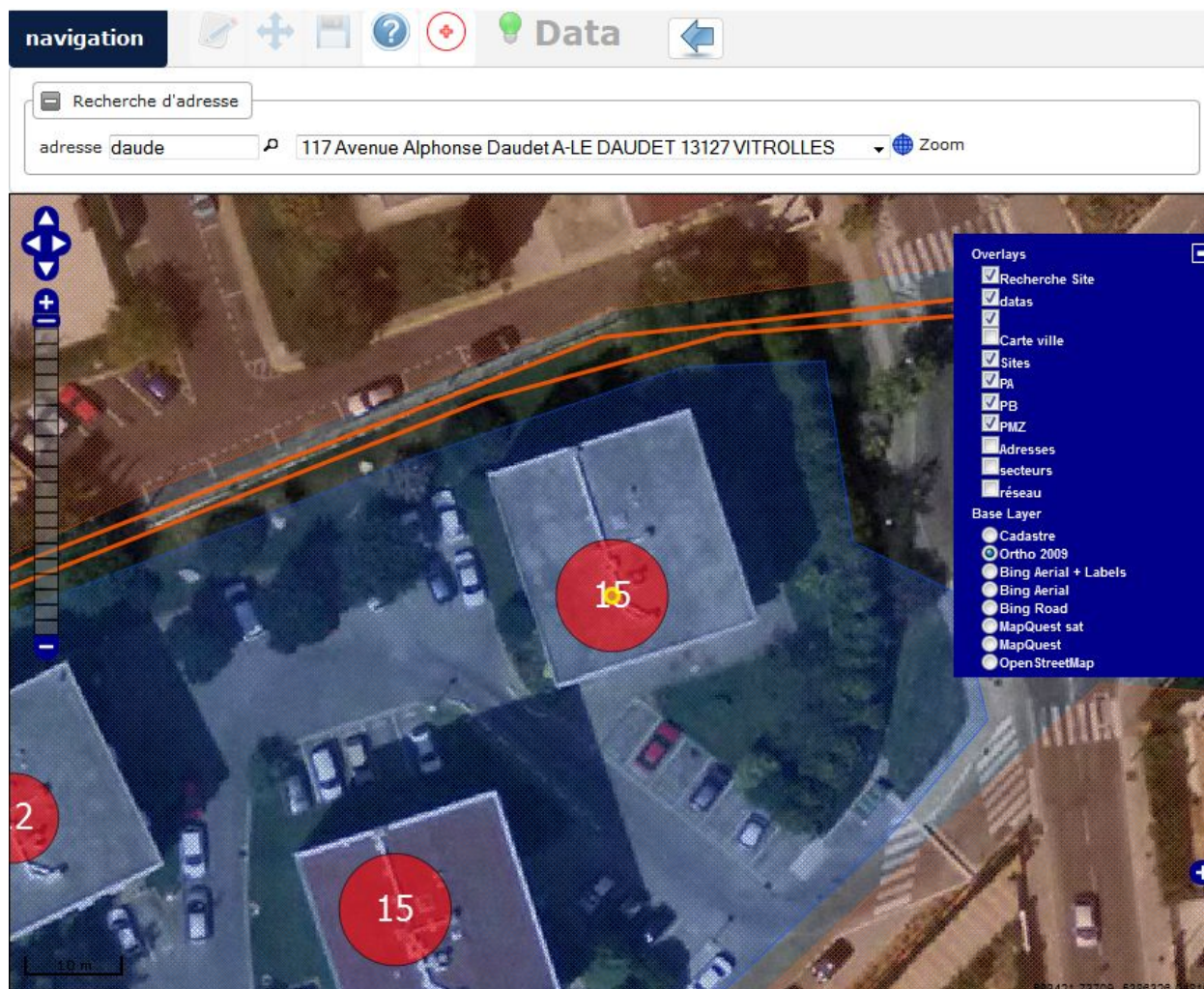
Overlays

- ☒ datas
- ☒ point
- ☒ Pluvial amont
- ☒ Pluvial aval
- ☒ Eclairage public
- ☒ Pluvial
- ☒ ERDF
- ☒ GRT Gaz
- ☒ Ruissellement
- ☒ Fibre optique
- ☒ Cadastre
- ☒ Adresse

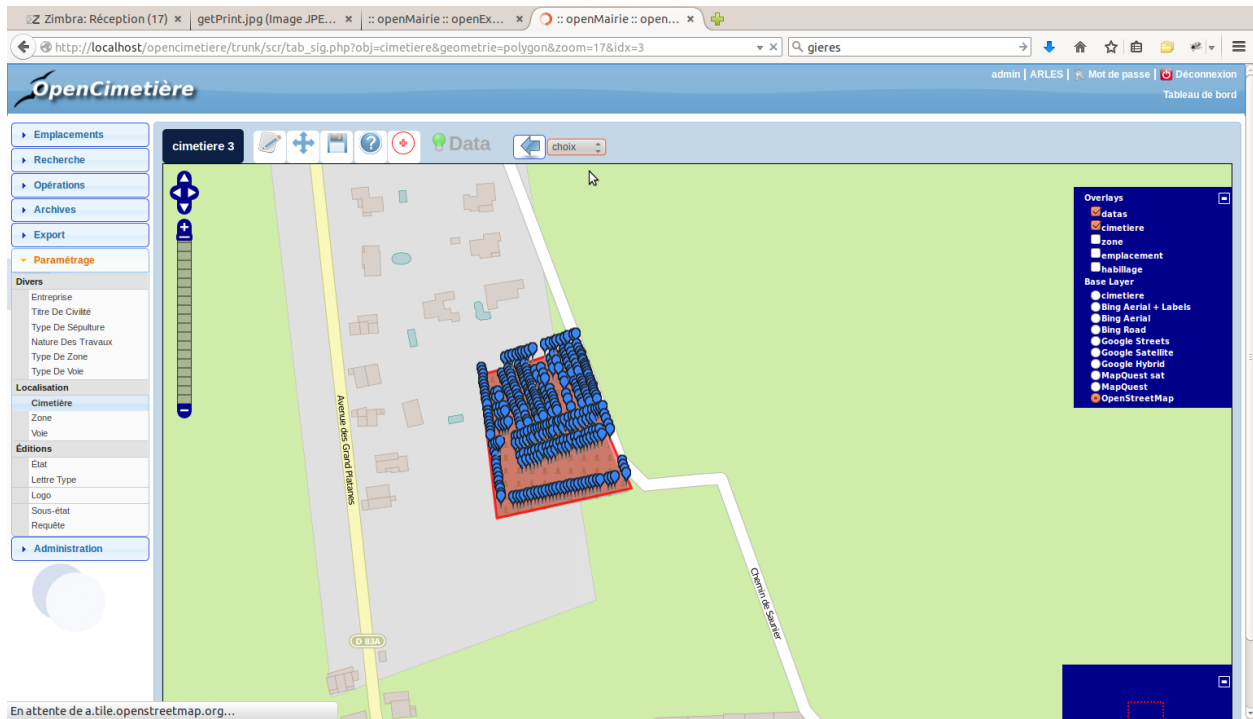
Base Layer

- ☒ Cadastre
- ☐ Photo 2009
- ☐ Bing Aerial + Labels
- ☐ Bing Aerial
- ☐ Bing Road
- ☐ MapQuest sat
- ☐ MapQuest
- ☐ OpenStreetMap

La barre permet une recherche d'adresse dans openFFTH



La barre permet de changer de cimetiere dans openCimetiere



3.2.3.2 Saisie des cartes :

Il est possible de lister les cartes disponibles dans le menu administration -> option om_sig_map

Administration → Om_sig_map										
om_sig_map										
1 - 2 enregistrement(s) sur 2										
	ident.	identifiant	libellé	act	idx	reqmo	rec.	src.flux	fond	etendue
	2 fils	Fils	Oui	Oui	Non	Non				12 Commune: Vitrolles (13117)
	1 pere	Père	Oui	Oui	Non	Oui				13 Commune: Vitrolles (13117)

3.2.3.2.1 Formulaire

Il est possible de modifier / supprimer les cartes dans le formulaire de saisie om_sig_map en appuyant sur modifier ou supprimer

The screenshot shows a web interface for configuring a map layer. At the top, there are tabs for 'om_sig_map', 'om_sig_map_comp', and 'om_sig_map_flux'. The main content area is titled 'om_sig_map * 1' and contains several sections:

- Retour**: A blue arrow icon and the text 'Retour'.
- titre**: A section with 'Identifiant *' set to 'pere' and 'libellé *' set to 'Pere'. There is an 'actif' checkbox.
- Cas d'utilisation**: A section with three options: 'à partir d'un enregistrement donné : Flux en provenance de la carte :', 'd'une requête mémorisée : Choisir source_flux', and 'd'une recherche :'. The first option is selected.
- fond**: A section with 'zoom *' set to '16' and 'Fond par défaut : *'. There are also fields for 'osm:' (set to '13'), 'Bing:', 'sat:', and 'layer_info'.
- Paramètres cartographiques**: A section with 'étendue *' (set to 'Commune: Vitrolles (13117)'), 'Restreindre la navigation sur l'étendue', 'Centrer sur le point (si différent du centre de l'étendue)', and 'projection *' (set to 'lambert93').
- marqueurs**: A section with 'URL *' (set to './scrform.php?obj=pere&idx=') and 'requête sql *' (containing a complex SQL query).
- SLD**: A section with 'Fichier SLD pour les marqueurs' and 'Fichier SLD pour les données (om_sig_map_comp)'. There are also 'Retour' buttons.

At the bottom, there is a button 'Modifier l'enregistrement de la table : om_sig_map' and a blue arrow icon with the text 'Retour'.

3.2.3.2.2 Description des champs :

L'id est un identifiant unique permettant de d'identifier la carte dans les formulaires de saisie ou d'affichage.

Il est noté que quand le formulaire de saisie est généré par le générateur, il porte automatiquement comme identifiant le nom de la table.

Les cas d'utilisation de la carte peuvent être : à partir d'un enregistrement (cas des formulaires), suite à une requête mémorisée (voir export), dans le cadre d'une recherche (cas affichage)

Source flux permet de récupérer les flux wms d'une carte om_sig_map "et d'éviter d'avoir à les resaisir.

Pour modifier une carte, il est possible de créer la carte avec actif = non (éventuellement par copie) et ensuite la rendre active "et désactiver l'ancienne). Cette méthode permet de retourner en arrière

Le zoom est le zoom d'affichage en fonction du centre

Les fonds externes affichés sont les suivants : OSM, Bing , Google ,sat

Il est possible d'afficher un om_sig_map_flux par défaut en donnant son numéro.

Il faut que le flux soit déclarer comme fond (???) dans om_sig_map_flux

On affiche un fond en indiquant

```
osm pour osm
sat pour google
bing pour bing
```

Le « layer info » est constitué de marqueurs issu de la requête SQL

L'étendue est la possibilité d'étendue par défaut de la carte

Il est possible de restreindre la navigation à l'étendue

Il est possible de centrer sur un point différent que le centre de l'étendue en utilisant la carte om_sig_map. (vérifier)

Les choix de projection sont définis dans dyn/var_sig.inc.php

Pour les marqueurs, l'url d'affichage de données est paramétré dans le champ url, la requête dans le champ requête sql.

Les marqueurs sont constitués à partir d'une requête et sont stockés dans un format geo json.

Les fichiers SLD pour les marqueurs (données json) et les données (données vecteurs) sont téléchargeables dans le module SLD dans le cadre du filesystem choisi.

Exemple de requête SQL pour affichage des marqueurs

```
SELECT  ST_asText(p.point_geom) AS geom,
        p.libelle AS titre,
        p.libelle AS description,
        p.pere AS idx,
        np.libelle AS nomenclature_lib,
        np.nomenclature_pere AS nomenclature_code
FROM    &DB_PREFIXE.pere p
        LEFT JOIN &DB_PREFIXE.nomenclature_pere np
        ON np.nomenclature_pere = p.nomenclature_pere
WHERE   p.pere IN (SELECT &idx::integer UNION &lst_idx)

-- variables
&DB_PREFIXE = shema
&idx = géométrie courante
&lst_idx = liste des géométries courantes

-- marqueur(s)
Père 01aa aa (1)  titre + idx
Père 01aa aa      description
nomenclature_lib: Nomenclature pere 02
nomenclature_code: 02
```

3.2.3.3 Saisie des flux :

Les flux permettent de créer des couches internes ou externes à l'application. Il est traité les flux wms (web map service) et les flux de tuiles (tiles)

Il est possible de lister les flux dans le menu administration -> option om_sig_flux

Administration ➤ Om_sig_flux				
om_sig_flux				
1 - 5 enregistrement(s) sur 5				
+ om_sig_flux				
	libelle	id	collectivite	type
	3 adresse	adresse	LIBREVILLE (1)	WMS
	4 cadastre	cadastre	LIBREVILLE (1)	WMS
	1 métier	metier	LIBREVILLE (1)	WMS
	2 métier filtré	metier_filtre	LIBREVILLE (1)	WMS
	5 Mini	Mini	LIBREVILLE (1)	Slippy Map Tiles

3.2.3.3.1 Formulaire

Il est possible de modifier / supprimer les flux dans le formulaire de saisie om_sig_flux en appuyant sur modifier ou supprimer

3.2.3.3.2 Description des champs :

- om_sig_flux est la clé primaire automatique
- id est l'identifiant unique du flux
- attribution : permet d'afficher sur la carte l'attribution des données (copyright ou copyleft)
- type de flux

```
wms (vide) : exemple qgis
tilecache (TCF) : tuiles
sleepy map tile (SMT)
impression (IMP) : envoi d'un getPrint ?
```

- URL : url du flux
- Couches : saisir les couches séparées par des virgules
- Paramètres varie suivant le type de flux

```
si c'est un flux wms = vide
si c'est une impression
    largeur carte dans composeur x 2 :
    hauteur carte dans composeur x 2
si c'est des tuiles (TCF ou SMT)
    URL pour GetFeatureInfo :
    couches pour GetFeatureInfo :
```

Exemple d'url de flux wms (sur linux avec qgis)

```
http://localhost/cgi-bin/qgis_mapserv.fcgi
?SERVICE=WMS&VERSION=1.3.0
&map=/var/www/openfoncier/trunk/app/qgis/openfoncier.qgs
```

Exemple d'une requête impression : getprint

Le getprint ne fonctionne que pour qgis.

https://hub.qgis.org/wiki/17/QGIS_Server_Tutorial

Administration ➤ Om_sig_wms ➤ 13 IMPRESSION A4 PAYSAGE

om_sig_wms 13

libelle : impression A4 paysage

id : A4Paysage

Type de flux : IMP

adresse

URL ([EXTENT], [LAYERS]) : `http://localhost/cartes/qgis/qgis_mapserv.fcgi.exe?SERVICE=WMS&VERSION=1.3.0&map=c:/osgeo4w/projetsqgis/openadresse/imp.qgs&REQUEST=GetPrint&TEMPLATE=A4PAYSAGE&map0:extent=[EXTENT]&SRS=EPSG%3A3857&WIDTH=210&HEIGHT=297&FORMAT=pdf&DPI=300&LAYERS=[LAYERS]`

couches (séparées par ,) : Cadastre, adresse

paramètres

largeur carte dans composeur x 2 : 574

hauteur carte dans composeur x 2 : 400

Modifier
Supprimer

3.2.3.4 Saisie des géométries :

om_sig_map_comp permet d'associer un ou plusieurs champs géométriques lié à la table correspondante au formulaire concernés.

Ces champs géométriques peuvent être mis à jour dans l'interface.

Ces champs constituent la couche vectorielle (modifiable) de la carte.

Il est possible de lister les géométries concernés dans le menu administration -> option om_sig_map onglet, om_sig_map_comp.

Le champ géométrique à mettre à jour se fait dans un sous formulaire d'om_sig_map

Administration ➤ Om_sig_map ➤ 1 PERE

om_sig_map om_sig_map om_sig_map_comp om_sig_map_flux

1 - 2 enregistrement(s) sur 2

	ident.	id.map	libellé	obj.	ord.	act.	maj.	req.
	6	pere	Point	pere	0	Oui	Oui	pere.pere(point_geom->point)
	1	pere	Périmètre	pere	2	Oui	Oui	pere.pere(perim_geom->multipolygon)

Retour

3.2.3.4.1 Formulaire

Il est possible de modifier / supprimer les géométries dans le sous formulaire de saisie om_sig_map_comp en appuyant sur modifier ou supprimer

Administration ➤ Om_sig_map ➤ 1 PERE

om_sig_map om_sig_map om_sig_map_comp om_sig_map_flux

administration ➤ om_sig_map_comp ➤ 6

Retour

om_sig_map_comp * 6
 om_sig_map * Père
 Nom géométrie : * Point
 Objet : * pere
 Ordre d'affichage : * 0
 Actif : ☐

mise-à-jour

Mis à jour :
 Table : pere Champ idx : pere Champ géographique : point_geom
 Type de géométrie : point

Modifier l'enregistrement de la table : 'om_sig_map_comp' Retour

Retour

3.2.3.4.2 Description des champs :

Le champ om_sig_map_comp est la clé primaire numérique automatique

om_sig_map est la clé primaire de la carte concernée

Il faut saisir le nom de la géométrie (champ géométrique) et l'objet concerné

Si la mise à jour est autorisée (vrai), il faut saisir

- la table du champ géométrique à modifier
- le nom de champ correspondant à la clé primaire de l'enregistrement à modifier
- le champ géographique à modifier
- le type de géométrie

3.2.3.5 Saisie des flux de la carte :

om_sig_map_flux permet d'associer un flux à une carte.

Il est possible de lister les flux d'une carte dans le menu administration -> option om_sig_map, onglet om_sig_map_flux.

Le flux associé est dans un sous formulaire d'om_sig_map

Administration ➤ Om_sig_map ➤ 1 PERE

om_sig_map om_sig_map om_sig_map_comp om_sig_map_flux

1 - 8 enregistrement(s) sur 8

Identifiant	om_sig_map	flux	nom ol	base	ordre	vis.	panier
10 Père		cadastre (WMS)	Panier - Bati	Non	1	Non	Oui
11 Père		cadastre (WMS)	Panier - Parcelle	Non	2	Non	Oui
6 Père		adresse (WMS)	Panier - Adresse	Non	3	Non	Oui
3 Père		métier filtré (WMS)	Métier filtré	Non	3	Oui	Non
9 Père		cadastre (WMS)	Cadastre	Oui	1	Non	Non
13 Père		Mini (SMT)	Orthophoto 2009	Oui	2	Non	Non
1 Père		métier (WMS)	Métier	Non	1	Oui	Non
5 Père		adresse (WMS)	Adresse	Non	2	Non	Non

Retour

3.2.3.5.1 Formulaire :

Il est possible de modifier / supprimer les fluxs dans le sous formulaire de saisie om_sig_map_flux en appuyant sur modifier ou supprimer

3.2.3.5.2 Description des champs :

- om_sig_map_flux est la clé primaire numérique
- flux wms permet d'associer un flux wms (om_sig_wms) à la carte (om_sig_map)
- le nom map openLayers sera le nom qui apparaîtra sur la carte dans l'onglet « couche » ou « fond »
- fond de carte permet d'associer le flux comme un fond
- ordre : permet de gérer l'ordre d'apparition des couches (ou fond) dans l'onglet correspondant
- visible par défaut : affiche la couche si c'est vrai à l'ouverture de la carte
- singletile : ramène le flux wms en une seule image pour la fenêtre et non en une série d'imagette, ce qui permet de résoudre le problème des étiquettes tronquées. ATTENTION les temps de réponses peuvent s'allonger car le cache (du serveur wms ou du navigateur ???) n'est pas utilisé
- panier : le panier permet de stocker des géométries en vue de définir la géométrie de l'objet en cours : exemple : un permis de construire est un ensemble de parcelles

En cas d'utilisation du panier, il faut :

- donner un nom au panier : exemple : panier parcelle
- désigner la couche (layer) à utiliser : parcelle
- désigner l'attribut à récupérer : ex : parcelle
- définir le caractère d'encapsulation (en général la “ si alpha numérique ??? et vide si numérique)
- construire la requête d'union qui va être la nouvelle géométrie :
- donner le type de géométrie : le plus souvent un polygone
- définir la requête de filtrage qui va permettre de filter le flux wms

exemple d'une requête construisant une géométrie faisant une union des parcelles sélectionnées dans le panier

```
select st_astext(st_union(geom)) as geom from &DB_PREFIXEparcelle
where parcelle in (&lst)
```

&DB_PREFIXE = schéma

&lst = liste des géométrie du panier

Exemple d'une requête filtrée

pour produire le filtre suivant :

```
layer1:"champ1" = 'valeur1',layer2:"champ2" = 'valeur2'
```

il faut entrer la requête suivante pour selectionner les electeurs d'un bureau :

```
select 'electeur:²bureau² = ''||bureau.bureau||'' as buffer
from &DB_PREFIXEbureau where bureau = '&idx'
```

```
select 'electeur:²bureau² = '&idx'' as buffer from &DB_PREFIXEbureau
where bureau = '&idx'
```

-- parametres

² = caractère utilisé pour les doubles quotes : "

|| concatenation sql

''' permet d echapper la simple quote

' sql remplace les deux quotes par une quote (caractere quote)

le filtre final appliqué au flux wms est : electeur:"bureau" = '04' pour le bureau 04

autre exemple le père et tous ses fils

```
SELECT 'fpere_point:²pere² IN ( '||pere||' );fpere_perim:²pere² IN ( '||pere||' );
```

```
↪ffils_point:²pere²
```

```
IN ( '||pere||' );ffils_perim:²pere² IN ( '||pere||' );ffils_perim:²pere² IN (
```

```
↪'||pere||' )'
```

```
AS buffer FROM ( SELECT array_to_string(array_agg(pere), ' , ') AS pere FROM &DB_
```

```
↪PREFIXEpere
```

```
WHERE pere IN (SELECT &idx::integer UNION &lst_idx) ) a
```

cette table permet de saisir les extends pour les cartes om_sig_map

3.2.3.6 Saisie om_sig_extent

Il est possible de lister les extents dans le menu administration -> option om_sig_extent

1 - 15 enregistrement(s) sur 39340			Page 1 / 2623		Tous	Recherche
	om_sig_extent	nom	extent	valide		
	1	Arrondissement: Abbeville (801)	1.38015228874262,49.8395709942243,2.10618821851606,50.3699899402347	Non		
	2	Arrondissement: Agen (471)	0.309009857616763,44.023997544888,0.950863045147848,44.3794671820947	Non		
	3	Arrondissement: Aix-en-Provence (131)	4.95408123114882,43.3681731692067,5.81352048404361,43.7487066717393	Non		
	4	Arrondissement: Ajaccio (2A1)	8.53471699546624,41.6990352706686,9.25212694255545,42.3821724312197	Non		
	5	Arrondissement: Albertville (731)	6.22010507832685,45.2518538652093,7.1147113983013,45.9094618505331	Non		
	6	Arrondissement: Albi (811)	1.53529654853871,43.7454568339649,2.58023117023956,44.2014794704187	Non		
	7	Arrondissement: Alençon (611)	-0.860552265531819,48.3752096839186,0.452765096577728,48.7063564916053	Non		

3.2.3.6.1 Formulaire

Il est possible de modifier / supprimer les om_sig_extents dans le formulaire de saisie om_sig_extent en appuyant sur modifier ou supprimer

exemple d'extent (libellé)

```
1.38015228874262,49.8395709942243,2.10618821851606,50.3699899402347
```

Ces 4 chiffres représentent les coordonnées (x,y) de 2 points haut-gauche et bas-droit d'un rectangle déterminant la portion de carte à représenter

Pour accéder aux extents dans om_sig_map, cochez la case « valide » dans le formulaire.

3.2.3.6.2 Les extent par défaut

Par défaut, il est proposé les extent (data/pgsql/om_extent.sql) :

- des régions françaises
- des départements
- des arrondissements
- des EPCI
- des communes françaises

Par défaut, Arles et Vitrolles sont valides.

3.2.3.6.3 Construction d'un extent avec openStreetMap :

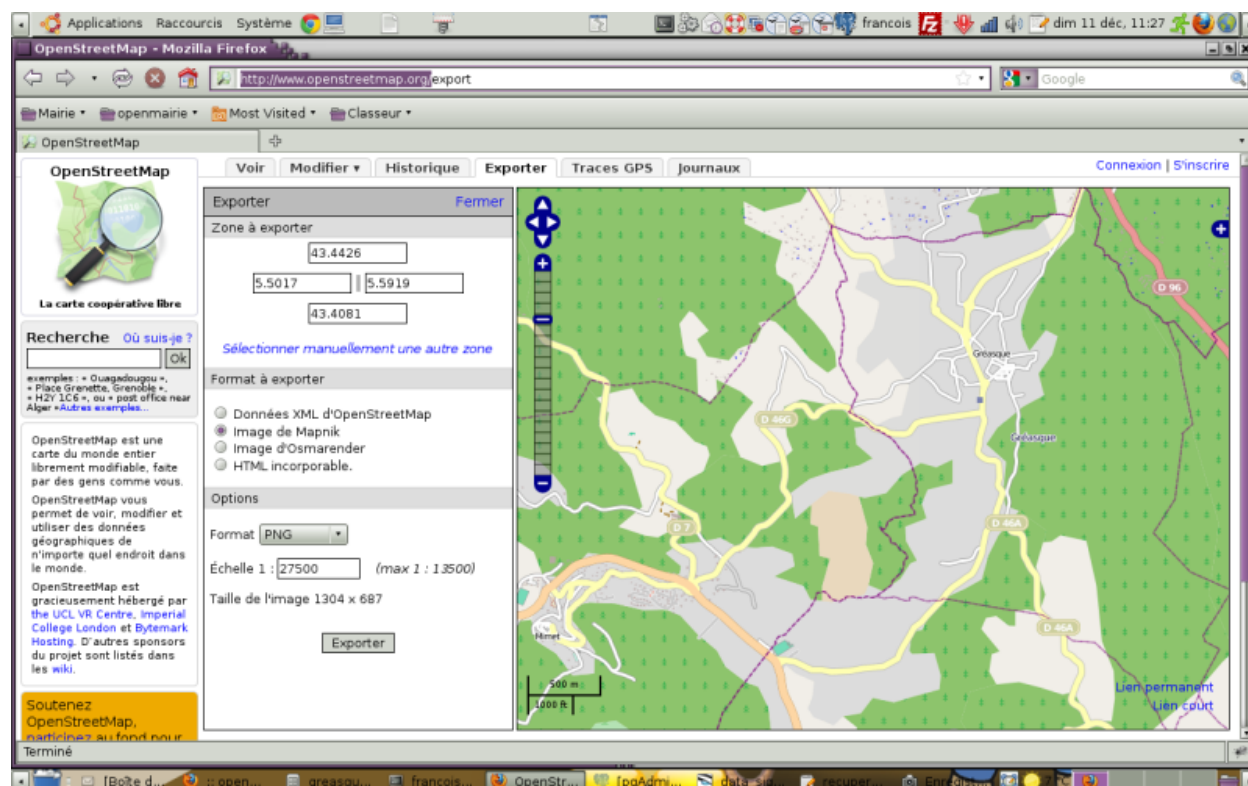
Pour adapter ses cartes à un autre périmètre

Aller sur openstreetMap <http://www.openstreetmap.org/>

Chercher la ville : exemple « Gréasque »

Ajuster la carte aux frontières communales

Aller dans l'onglet export et noter les coordonnées géographiques « zone à exporter »



3.3 Génération

Précision sur le vocabulaire utilisé dans cette documentation.

Modèle de données

« En informatique, un modèle de données est un modèle qui décrit de façon abstraite comment sont représentées les données dans une organisation métier, un système d'information ou une base de données. »

- cf. Wikipédia, Article [Modèle de données](#).

Dans la documentation suivante, le terme modèle de données est utilisé pour désigner les classes métier d'openMairie ainsi que les formulaires qu'elles représentent.

Objet

Le mot objet fait référence aux instances des classes d'openMairie et, par extension, aux enregistrements en base de données qui les représentent.

3.3.1 Introduction

Le générateur permet de construire des applications à partir de l'analyse d'un schéma d'une base de données.

Les informations récupérées dans le schéma sont les suivantes :

- la liste des tables ;
- le nom, le type et les contraintes de chaque colonne.

Sur cette analyse le générateur crée les modèles de données. openMairie gère :

- les clés primaires mono-colonne ;
- les clés étrangères ;

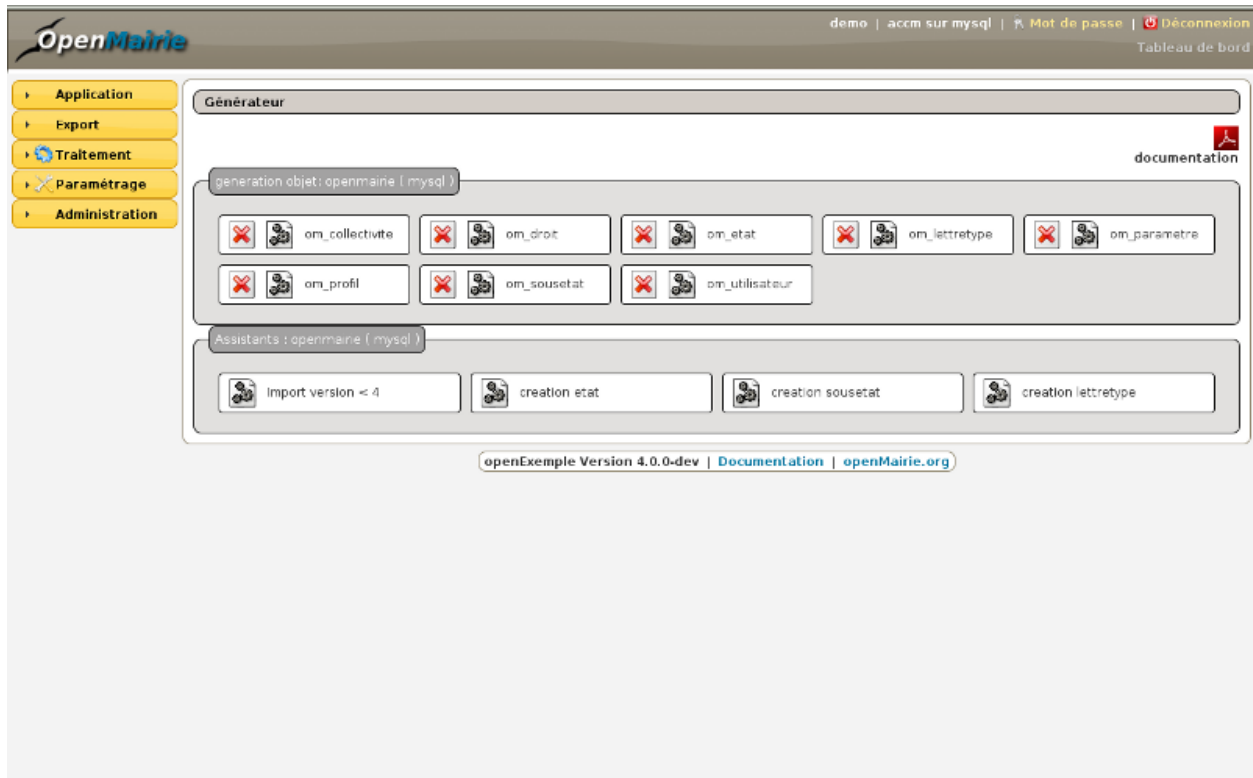
— la multi-collectivité (selon la présence d'un champ `om_collectivite`).

Le générateur contient également des assistants permettant de créer facilement des états associés à des collectivités.

Attention : la version mysql est abandonnée

3.3.2 L'interface

Le menu generateur est le suivant :



En appuyant sur la touche generation on accède à l'écran de génération qui se décompose en :

- une analyse de la base de donnée en cours et de la table choisie
- un état des fichiers existants ou non
- les options de génération

OpenMairie

demo | accm sur mysql | Mot de passe | Déconnexion

Tableau de bord

Application

Export

Traitement

Paramétrage

Administration

Générateur Gen

analyse bdd mysql openmairie

table bdd [om_collectivite | om_droit | om_etat | om_lettretype | om_profil | om_sousetat | om_utilisateur]

om_parametre [cle N - cle automatique | longueur enregistrement : 92]

champ [om_parametre 11 int | libelle 20 string | valeur 50 string | om_collectivite 11 int]

sousformulaire

classe secondaire [om_collectivite]

choix paramétrage standard

generation formulaire

☒ table.inc om_parametre.inc.php sql/mysql/om_parametre.inc.php existant

☐ table.inc om_parametre.inc ../sql/mysql/om_parametre.inc existant

☒ tableform.inc om_parametre.form.inc.php sql/mysql/om_parametre.form.inc.php existant

☐ tableform.inc om_parametre.form.inc ../sql/mysql/om_parametre.form.inc existant

☒ obj om_parametre.class.php obj/om_parametre.class.php existant

☐ obj om_parametre.class.php ../obj/om_parametre.class.php existant

generation edition

☐ om_parametre.pdf.inc ../sql/mysql/om_parametre.pdf.inc non existant

generation reqmo

☐ om_parametre.reqmo.inc ../sql/mysql/om_parametre.reqmo.inc existant

☐ om_parametre_om_collectivite.reqmo.inc ../sql/mysql/om_parametre_om_collectivite.reqmo.inc non existant

generation divers

☐ ../sql/mysql/om_parametre.import.inc ../sql/mysql/om_parametre.import.inc existant

valider generation om_parametre

openExemple Version 4.0.0-dev | Documentation | openMairie.org

3.3.2.1 Analyse de la base

Le programme propose une analyse de la base en cours :

- liste des tables de la base
- l'information sur la clé primaire de la table
- la longueur de l'enregistrement de la table
- les informations sur les champs : nom, type et longueur
- les clés secondaires (exemple table om_collectivite)
- les sous formulaires à associer

A partir de la version 4.2.0, il n'y a plus de choix de paramétrage dans l'écran.

3.3.2.2 Les fichiers à générer

Il est proposé une liste de case à cocher :

La case est cochée sur le fichier correspondant n'existe pas (colonne de droite)

Le formulaire métier auto généré, table.inc, tableform.inc est toujours coché (fichiers en gen/) :

- gen/obj/table.class.php
- gen/sql/basededonnees/table.inc
- gen/sql/basededonnees/table.form.inc

La génération de ces 3 fichiers ne met pas en péril votre programmation qui est en :

- obj/table.class.php
- sql/basededonnees/table.inc
- sql/basededonnees/table.form.inc

basededonnees = pgsql

3.3.3 Conditions de génération

3.3.3.1 Contraintes de la base de données

Pour qu'un modèle de données puisse être généré, il faut que la table de base de données qui le représente remplisse les conditions suivantes :

- la table doit avoir une clé primaire composée d'une seule colonne, ou une colonne portant le même nom que la table ;
- les clés étrangères doivent référencer des tables remplissant la condition ci-dessus.

Si l'une de ces conditions n'est pas satisfaite, les interfaces de génération affichent une erreur.

3.3.3.2 Contraintes du système de fichiers

Le générateur crée les classes métier de l'application ainsi que les fichiers de surcharge. Pour pouvoir créer ces fichiers, le serveur web PHP doit avoir les droits d'écriture dans les dossiers suivants :

- gen/obj
- gen/sql/pgsql
- gen/sql/mysql
- obj/
- sql/pgsql
- sql/mysql

Si des droits sont manquants :

- les scripts `genauto` et `gensup` ne permettent pas de générer ces fichiers que le serveur ne peut pas écrire ;
- le script `genfull` quant-à lui, affiche des messages après la génération indiquant quels fichiers ne sont pas accessibles (ces fichiers ne sont, bien entendu, pas générés).

3.3.4 Définition des modèles de données

3.3.4.1 L'identifiant

Chaque modèle de données doit avoir un champ destiné à contenir l'identifiant des objets. Sans ce champ, il n'est pas possible de créer un modèle.

3.3.4.1.1 Définition de l'identifiant

Il suffit d'ajouter la contrainte `SQL PRIMARY KEY` à une colonne d'une table pour créer un champ identifiant. Il sera ensuite automatiquement géré par openMairie lors de l'ajout, la modification et la suppression d'enregistrements.

3.3.4.1.2 Fonctionnement interne du générateur

Comment ce champ est déterminé lors de la génération d'un modèle ?

Le générateur suit la procédure suivante :

- il utilise la colonne ayant la contrainte `PRIMARY KEY` si elle existe ;
- sinon, il utilise la colonne ayant le même nom que la table.

S'il n'existe ni contrainte, ni colonne ayant le même nom que la table, le modèle ne peut pas être créé.

Note : Le fait d'utiliser une colonne ayant le même nom que la table pour déterminer le champ identifiant est là pour une raison de rétro-compatibilité. Les versions d'openMairie antérieures à 4.3.0 n'utilisaient pas encore la contrainte PRIMARY KEY.

il n'est pas possible d'utiliser une clé primaire composée de plusieurs colonnes

3.3.4.2 Les références vers d'autres objets

Un modèle de données peut contenir, un ou plusieurs champs, faisant référence à d'autres objets. Ces objets pouvant être de modèle différent.

3.3.4.2.1 Définition des références

La méthode pour créer des références diffère en fonction du SGBD.

3.3.4.2.1.1 Avec PostgreSQL

Il suffit d'ajouter la contrainte SQL FOREIGN KEY à des colonnes pour créer des champs de type référence.

3.3.4.2.2 Fonctionnement interne du générateur

Comment ces champs sont déterminés lors de la génération d'un modèle ?

Le générateur conserve une liste des colonnes qui donneront, après génération, des champs références. Pour créer cette liste, il suit la procédure suivante :

- avec PostgreSQL, le générateur peut interroger les tables du système contenant la liste des clés étrangères d'une table particulière, ainsi que les tables étrangères référencées par ces clés ;
- dans un second temps, indifféremment du SGBD, il ajoute à la liste des clés étrangères le nom des colonnes portant le même nom que d'autres tables.

La liste ainsi formée permettra au générateur de créer des champs de type référence dans les modèles de données.

3.3.4.2.3 Affichage dans les formulaires

Comment ces champs sont représentés dans les formulaires ?

Depuis le formulaire de l'objet faisant référence, ces champs sont représentés par des balises HTML `<select>`. L'ensemble des objets pouvant être référencés sont listés sous la forme d'options.

Depuis le formulaire de l'objet référencé, un onglet apparaît pour chaque modèle différent faisant référence à cet objet. Chaque onglet liste l'ensemble des objets faisant référence à l'objet présenté en formulaire.

3.3.4.3 Les champs uniques

Un modèle de données peut contenir un ou plusieurs champs uniques. Il n'est pas possible pour plusieurs objets d'un même modèle d'avoir la même valeur pour ce champ.

Un modèle de données généré peut également contenir, au plus, un groupe de champs unique. Cette fois, c'est la combinaison des valeurs de ces champs qui ne pourra exister qu'une seule fois.

3.3.4.3.1 Définition des champs uniques

Il suffit de définir une contrainte SQL `UNIQUE` sur une colonne ou un groupe de colonnes pour créer respectivement un ou plusieurs champs uniques.

3.3.4.3.2 Fonctionnement interne du générateur

Comment ces champs sont déterminés lors de la génération d'un modèle ?

L'abstracteur de base de données d'openMairie peut, en analysant une table, récupérer la liste de ses colonnes uniques.

3.3.4.3.3 Affichage dans les formulaires

Comment ces champs sont représentés dans les formulaires ?

Ces champs sont affichés indifféremment des champs sans contrainte.

Lors de la validation d'un formulaire, une vérification est faite pour chaque champ unique, ainsi que pour un éventuel groupe de champs uniques. Si une valeur (ou combinaison) est déjà présente dans la base de données, un message d'erreur est affiché, et la base de données n'est pas modifiée.

3.3.4.4 Les champs requis

Un modèle de données peut contenir un ou plusieurs champs requis.

3.3.4.4.1 Définition des champs requis

Il suffit de définir une contrainte SQL `NOT NULL` sans clause `DEFAULT` sur une colonne pour créer un champ requis.

Attention : En ajoutant une clause `DEFAULT` à une contrainte `NOT NULL` nous indiquons clairement au générateur que **le champ n'est pas requis** ! La valeur par défaut permet à l'utilisateur de laisser le champ vide lors d'une validation de formulaire. Le SGBD se charge alors d'ajouter lui même cette valeur.

3.3.4.4.2 Fonctionnement interne du générateur

Comment ces champs sont déterminés lors de la génération d'un modèle ?

L'abstracteur de base de données d'openMairie peut, en analysant une table, récupérer la liste de ses colonnes requises n'ayant pas de valeur par défaut.

3.3.4.4.3 Affichage dans les formulaires

Comment ces champs sont représentés dans les formulaires ?

Ces champs sont affichés avec un marqueur à côté de leur libellé, indiquant qu'ils sont requis. Par défaut openMairie utilise le caractère `*` pour indiquer les champs requis.

Si ces champs ne sont pas remplis lors de la validation d'un formulaire, un message d'erreur est affiché pour chaque champ requis non complété, et la base de données n'est pas modifiée.

3.3.4.5 Le champ libellé

Pour représenter des objets dans des champs de type `<select>`, le générateur utilise un champ textuel particulier appelé libellé.

Ce champ est également utilisé pour ordonner les éléments d'un tableau de manière croissante.

3.3.4.5.1 Définition du libellé

Pour définir explicitement une colonne comme libellé d'un modèle, il faut la nommer `libelle`.

Si cette colonne n'existe pas, le générateur considère la deuxième colonne de la table comme étant un libellé (ce système était celui utilisé dans les versions d'openMairie 4.2.0 et inférieures).

Enfin s'il n'existe pas de seconde colonne, la clé primaire de la table est utilisé.

3.3.5 Fonctionnalités avancées

3.3.5.1 Ajouter une date de validité à un modèle

3.3.5.1.1 Description

Le générateur permet de créer des objets qui seront considérés comme valides seulement pendant une période donnée.

Qu'est ce qu'un objet à date de validité ?

Un objet à date de validité est un objet contenant deux champs spécifiques :

- `om_validite_debut`, déterminant la date de début de validité;
- `om_validite_fin`, déterminant la date de fin de validité.

Un objet valide se comporte comme un objet « traditionnel ». Par contre, lorsqu'il arrive à expiration, l'objet n'apparaît plus dans les tableaux, les sous-tableaux et les champs de sélection (sauf s'il est actuellement valeur de l'un de ses champs).

Un tel objet est valide lorsque :

- sa date de début de validité est nulle ET (sa date de fin de validité est nulle OU sa date de fin de validité est strictement supérieure à la date actuelle) OU,
- sa date de début de validité est inférieure ou égale à la date actuelle ET (sa date de fin de validité est nulle OU sa date de fin de validité est strictement supérieure à la date actuelle).

A l'inverse, il est considéré comme non-valide (expiré) lorsque :

- il n'est pas valide.

Est-il possible de consulter la liste des objets expirés d'un modèle donné ?

Oui.

Le tableau du modèle dispose d'un bouton `Afficher les éléments expirés` permettant d'afficher, en plus des objets valides, les objets expirés. Lorsque les éléments expirés sont affichés, bouton devient `Masquer les éléments expirés`.

3.3.5.1.2 Définition des dates de validité

Pour que le générateur considère un modèle comme « à date de validité », il faut que sa table de base de données contienne les deux colonnes suivantes :

- `om_validite_debut` DATE;
- `om_validite_fin` DATE.

Important : Il ne faut surtout pas définir de contrainte NULL ou DEFAULT sur ces deux colonnes, sinon ces champs seront obligatoires à chaque validation de formulaire.

3.3.5.1.3 Affichage dans les formulaires

Ces champs apparaissent dans les formulaires sous la forme de `datepicker`, comme des champs de type date classiques.

3.3.6 L'analyse de la base

Les informations de la base sont analysées par la méthode « constructeur » de `gen.class.php`.

La construction des formulaires se fait suivant 5 types de champs reconnus par le générateur :

```
- string : chaîne de caractère
- int : nombre (entier)
- float : nombre decimal
- date
- blob : texte
- geom : geometry (pour postgres)
```

3.3.6.1 Type de champs

la champ String est du type openMairie (méthode `setType()`) :

- text dans le cas général
- `hiddenstatic` si modification pour clé primaire
- `select` pour clé secondaire

Le champ date est du type openMairie date avec calendrier et java script de contrôle de saisie de date

(La date est au format français JJ/MM/AAAA)

le champ Int est du type openMairie (methode `setType()`)

- `hidden` si clé primaire en ajout
- `hiddenstatic` si clé primaire en modification
- text avec contrôle numérique en javascript
- `select` pour clé secondaire

Le champ Blob est du type openMairie `textarea`

La longueur et la largeur sont définis en fichier de paramétrage `form.inc`

La taille n est pas pris en compte dans la longueur d'enregistrement

Les paramètres de `dyn/form.inc` permettent d'établir la longueur et la largeur d'affichage d'un blob :

```
$max=6; // nombre de ligne blob
$taille=80; // taille du blob
```

Les champs de type `geometry` sont des champs `geom` (accès a la fenetre `tab_sig.php`)

3.3.6.2 Equivalence type pgsql / type openMairie

L'information fournie par postgresql est moins complète que celle de mysql surtout au niveau de la longueur des champs « string » où il est fourni : la longueur de stockage qui est égal à -1 quand le stockage est variable type pgsql (longueur) type tableinfo si différent -> type openMairie

Bigint	(8)	int8	-> int
Smallint	(2)	Int2	-> Int
Integer	(4)	Int4	-> Int
Real	(4)	Float4	-> float
Doubleprecision	(8)	Float8	-> float
Numeric	(20)	Numeric	-> float
Money	(8)	Money	-> float
Char	(1)	Char	-> String (Quelque soit la longueur= 1)
Character	(-1)	Bpchar	-> String (Utilisation de la longueur d
			↪ 'affichage)
Character varying	(-1)	Varchar	-> String (Utilisation de la longueur d
			↪ 'affichage)
Text	(-1)	text	-> blob (Utilisation des paramètres de
			↪ form.inc)
Date	(4)	Date	-> Date (Utilisation des paramètres de
			↪ form.inc -
			\$pgsql_longueur_date)
geometry	-5		-> geom
boleen		boolean	-> checkbox

Pour postgresql, il est proposé dans form.inc 2 variables qui sont avec la version 4.2.0 inutiles car les longueurs sont gérées par le générateur (valeurs négatives)

```
$pgsql_taille_default = 20; // taille du champ par défaut si retour pg_field_prtlen =0
$pgsql_taille_minimum = 10; // taille minimum d affichage d un champ
```

Attention, pour les champs geom, il faut gérer la carte à chercher pour l'affichage de la carte en fenêtre

```
exemple de surcharge de la méthode setSelect pour afficher la carte dossier (de la
↪ table om_sig_map)

if($maj==1){ //modification
    $contenu=array();
    $contenu[0]=array("dossier",$this->getParameter("idx"));
    $form->setSelect('geom',$contenu);
}
```

3.3.6.3 Nom de champ et nom de table

Attention au nom de tables ou de champs, évitez les termes SQL : match, table, index, type, len ... ou openMairie : objet pour les noms de champs ou table.

Limites du generateur : Attention de ne pas utiliser les majuscules dans les noms de tables ou champs.

postgresql réagit assez mal avec les majuscules qu'il met entre guillemets dans les requêtes.

3.3.7 Les fichiers générés

Les fichiers générés concernent :

- les formulaires

- les requêtes mémorisées
- le script d'import de données

3.3.7.1 Formulaires

Les formulaires sont générés suivant le nom de la table dans le répertoire sql, sous repertoire portant le nom de la base pour régler le problème de compatibilité SQL (concaténation, extraction ...)

Deux types de formulaire sont générés : type table, type form.

3.3.7.1.1 Paramètres de type table

- gen/sql/basededonnees/nom_table.inc
- sql/basededonnees/nom_table.inc

Par défaut :

- tri en affichage vide
- champ de recherche avec les champs string
- pas d'affichage de champ blog
- rattachement de sous formulaire
- affichage de l'édition de la table

Dans le fichier paramètres : form.inc

\$serie = nombre d'enregistrement par page

\$ico = icône par défaut

3.3.7.1.2 Paramètres de type Form

gen/sql/basededonnees/nom_table.form.inc

sql/basededonnees/nom_table.form.inc

Dans le fichier paramètres : form.inc

\$ico = icône par défaut

Par défaut :

- tous les champs sont affichés les uns en dessous des autres

3.3.7.2 Objets « métier »

L'objet métier généré est stocké en gen/obj/nom_table.class.php. Ce script ne doit pas être modifié car il est reconstitué à chaque génération :

Cela permet de pouvoir modifier la base de données (ajout, modification ou suppression de champs) et de régénérer tout ou partie de l'application

Un second script héritant de l'objet généré permet de surcharger les méthodes et de personnaliser l'objet métier.

Toutes les modifications doivent être faites dans ce script soit en héritant de la méthode, soit en surchargeant la méthode.

L'objet à personnaliser est stocké en obj/nom_table.class.php

Les méthodes générés dans l'objet métier gen/obj/nom_table.class.php sont par défaut les suivantes.

Le type de champs est :

- caché (hidden) en ajout pour la clé primaire automatique,
- modifiable en ajout si la clé primaire n'est pas automatique
- l'unicité de la clé primaire est vérifiée si elle est modifiable (version 4.2.0)
- la clé primaire est visible sans possibilité de modifier en modification
- la clé secondaire n'est pas modifiable en sous formulaire si c'est la clé primaire du formulaire
- la clé secondaire est un champ select qui reprend les informations de la table liée
- la date est au format français
- geom si ce champ est géométrique (version 4.2.0)

La longueur d'affichage et le maximum autorisé à la saisie est celle contenu dans la base d'origine

Le contrôle des clés secondaires des autres tables est généré : il n'est pas possible de supprimer un enregistrement si des enregistrements sont liés à la clé primaire

Il est vérifier l'unicité de la clé si elle n'est pas automatique (version 4.2.0). Les libellés sont les noms des champs.

Ce module sert pour le formulaire et le(s) sous formulaire(s).

Les méthodes qui peuvent être implémentés dans `obj/nom_table.class.php` sont les suivantes

```
- verifier
- regroupe et groupe pour modifier les présentations [deprecated] utiliser setLayout
- trigger avant ou après l'enregistrement:
- triggerajouter
- triggermodifier
- triggersupprimer
- triggerajouterapres
- triggermodifierapres
- triggersupprimerapres
```

Les méthodes de l'objet généré en `gen/obj` peuvent être surchargées totalement ou partiellement :

Exemple

```
om_profil.class.php :
    surcharge des méthodes
        setValFAjout setId,
        verifierAjout
        et setType car la clé primaire est numérique et non automatique

om_utilisateur.class.php :
    champ pwd pour mot de passe methode partiellement surchargées (parent::setvalF(
    ↪$val);)
    setvalF, setType, setValsousformulaire,
    surcharge avec un javascript de mise en majuscule du nom
```

Enfin, il est possible de mettre en place d'autres type de champs disponible dans openMairie en surchargeant la méthode `setType` :

```
- ComboG combo gauche
- comboD combo droit
- Localisation (geolocalisation en x, y)
- http (lien)
- httpclick (lien)
- Password (Mot de passe)
- Pagehtml (Textearea pour affichage html)
- Textdisabled (Text non modifiable)
- Selectdisabled (Select non modifiable)
- Textreadonly (Text non modifiable)
- Hidden (champ caché)
```

(suite sur la page suivante)

(suite de la page précédente)

- Checkbox (case a cocher oui/non)
- Upload (chargement d'un fichier)
- voir (voir un fichier téléchargé)
- Rvb (choisir une couleur rvn avec la Palette de couleur) ...

voir framework/formulaire

3.3.7.3 Etats

Seul l'état « pdf » est généré par le générateur

Dans le menu gen (generateur), les états sont générés automatiquement avec un assistant.

Cet assistant vous permet de construire un état :

- en choisissant une table de la base
- en choisissant les champs à mettre dans l'état

L'état est enregistré dans la table om_etat et peut être modifié menu->administration -> etat

De la même manière, il est possible de créer un sous etat.

Il est possible de choisir le champ qui sera la clé secondaire en lien avec la table mère

Le sousetat est enregistré dans la table om_sousetat et peut être modifié

menu->administration -> sousetat

Le calcul de la largeur des colonnes est automatique dans les sous états et l'état pdf.

Attention : les champs « blob » ne sont pas pris en compte dans les éditions.

3.3.7.4 Requêtes mémorisées

Les requêtes paramétrées sont créées suivant le principe suivant :

- une requête globale
- une requête avec un champ select pour chaque clé secondaire (il est possible de sélectionner la requête à générer)
- Les autres champs sont sélectionnés à l'affichage

Les requêtes sont accessibles dans l'option du menu -> export.

3.3.7.5 Imports

Un script d'import des données est généré suivant le principe suivant :

- si la clé est automatique, génération du compteur
- tous les champs sont importés
- vérification de l'existence de la clé secondaire à chaque enregistrement

Les tables avec clés secondaires doivent donc être importées en dernier.

3.3.7.6 Mots-clefs Robot Framework

Un fichier de ressources de mots-clefs est créé pour chaque objet. De plus tous ces fichiers sont inclus dans un fichier ressource général.

(Pour plus d'informations voir [Génération](#).)

3.3.8 Paramétrage générateur

Le générateur fonctionne de manière autonome sans fichier de paramétrage. Il est tout de même possible de paramétrer certains éléments grâce à des fichiers de paramétrage déposés dans le répertoire `gen/dyn/`.

Il n'est pas nécessaire de personnaliser toutes les variables du fichier. Il est recommandé de déclarer uniquement les paramètres souhaités. Par défaut, le générateur prend les paramètres inclus dans la classe `gen`.

3.3.8.1 `gen/dyn/gen.inc.php`

Il permet de définir des paramètres généraux pouvant être utilisés partout dans le générateur.

```
<?php
/**
 * Mode de génération pour la gestion des identifiants et des références
 *
 * Permet de choisir par quel moyen sont récupérées les clés primaires et les
 * clés étrangères :
 * - "constraints" => en interrogeant les contraintes de la base de données
 * - "postulate" => par les postulats :
 *   - "le nom d'un champ 'clé primaire' a pour nom le nom de la table."
 *   - "le nom d'un champ 'clé étrangère' a pour nom le nom de la table vers
 *     laquelle elle fait référence, et fait référence au champ clé primaire
 *     de cette table."
 *
 * Default : $key_constraints_mode = "constraints";
 */
$key_constraints_mode = "postulate";

/**
 * Liste des tables à ne pas générer
 *
 * Permet de lister les tables dont la génération n'est pas souhaitable. Ces
 * tables n'apparaissent donc plus dans le menu de génération ni dans la
 * génération complète.
 *
 * Default : $tables_to_avoid = array();
 */
$tables_to_avoid = array(
    "om_version",
    "spatial_ref_sys",
);

/**
 * Ce tableau de configuration permet de donner des informations de surcharges
 * sur certains objets pour qu'elles soient prises en compte par le générateur.
 *
 * $tables_to_overload = array(
 *   "<table>" => array(
 *     // définition de la liste des classes qui surchargent la classe
 *     // <table> pour que le générateur puisse générer ces surcharges
 *     // et les inclure dans les tests de sous formulaire
 *     "extended_class" => array("<classe_surcharge_1_de_table>", ),
 *     // définition de la liste des champs à afficher dans l'affichage du
 *     // tableau champAffiche dans <table>.inc.php

```

(suite sur la page suivante)

(suite de la page précédente)

```

*      "displayed_fields_in_tableinc" => array("<champ_1>", ),
*      // définition des composantes du titre de la page : 'tablename' est
*      // la dernière composante du titre et 'breadcrumb' est la liste des
*      // premières composantes. Ce sont les chaines à traduire qui doivent
*      // être saisies ici. Le résultat serait :
*      // $ent = _("<libelle_1>")." -> "._("<libelle_2>")." -> "._("<libelle>");
*      "breadcrumb_in_page_title" => array("<libelle_1>", "<libelle_2>", ),
*      "tablename_in_page_title" => "<libelle>",
*      // désactivation des sousformulaire pour que les onglets ne s'affichent
*      // pas dans le contexte de la table
*      "tabs_in_form" => false,
*      // définition de l'option 'om_validite' pour que les colonnes soient
*      // cachées par défaut
*      "om_validite" => "hidden_by_default",
*  ),
* );
*/
$tables_to_overload = array(
    //
    "om_widget" => array(
        //
        "displayed_fields_in_tableinc" => array(
            "libelle", "om_profil", "type",
        ),
        //
        "breadcrumb_in_page_title" => array("administration", "tableaux de bord", ),
        "tablename_in_page_title" => "widgets",
        //
        "tabs_in_form" => false,
        //
        "om_validite" => "hidden_by_default",
    ),
);
?>

```

3.3.8.2 gen/dyn/tab.inc.php

Ce fichier n'est plus utilisé par le générateur depuis la version 4.5, les paramètres gérés dans ce dernier ont été transférés dans le fichier gen/dyn/gen.inc.php.

3.3.8.3 gen/dyn/form.inc.php

Ce script permet de personnaliser les éditions générées. On peut par exemple générer toutes les éditions au format A3.

Voici les variables personnalisables :

```

<?php
/**
 * Nombre d'enregistrements par page dans les listings
 */
$serie = 15;

```

(suite sur la page suivante)

(suite de la page précédente)

```

/**
 * Icône utilisée auparavant comme lien vers l'aide
 * @deprecated
 */
$ico = "../img/ico_application.png";

/**
 * Taille d'affichage du champ text (nombre de lignes)
 */
$max = 6;

/**
 * Taille d'affichage du champ text (nombre de colonnes)
 */
$taille = 80;

/**
 * Taille d'affichage du champ par défaut dans le cas où nous sommes
 * dans l'impossibilité de déterminer la taille du champ.
 * Uniquement pour le SGBD PostGreSQL
 */
$pgsql_taille_defaut = 20;

/**
 * Taille d'affichage du champ minimum pour ne pas afficher des
 * champs trop petits où la saisie serait impossible
 * Uniquement pour le SGBD PostGreSQL
 */
$pgsql_taille_minimum = 10;

/**
 * Taille d'affichage du champ maximum pour ne pas afficher des
 * champs trop grands où le formulaire dépasserait de l'écran
 * Uniquement pour le SGBD PostGreSQL
 */
$pgsql_taille_maximum = 30;

/**
 * Taille d'affichage de la date
 * Uniquement pour le SGBD PostGreSQL
 */
$pgsql_longueur_date = 12;

?>

```

3.3.8.4 gen/dyn/permissions.inc.php

Ce script permet de paramétrer la génération des permissions. L'objectif ici est de pouvoir indiquer des scripts à ne pas examiner et des permissions à ajouter à celles trouvées automatiquement.

Voici les paramètres disponibles :

```

<?php

/**

```

(suite sur la page suivante)

(suite de la page précédente)

```

* Liste des fichiers à ne pas prendre en compte
*
* Permet de lister les fichiers du répertoire obj/ dans lequel le système
* de génération des permissions ne doit pas passer.
*
* Default : $files_to_avoid = array();
*/
$files_to_avoid = array(
    "pdf_lettre_rar.class.php",
    "pilotage.class.php"
);

/**
* Liste des permissions spécifiques
*
* Permet de lister les permission que le système de génération des permissions
* n'est pas en mesure de trouver.
*
* Default : $permissions = array();
*/
$permissions = array(
    "proces_verbal_fichier_telecharger",
    "dossier_instructeur_modifier_instructeur",
);

?>

```

3.3.8.5 gen/dyn/pdf.inc.php

Ce script permet de personnaliser les éditions générées. On peut par exemple générer toutes les éditions au format A3.

Voici les variables personnalisables :

```

<?php

$longueurtableau = 280;
$orientation='L'; // orientation P-> portrait L->paysage";
$format='A4'; // format A3 A4 A5;
$police='arial';
$margeleft=10; // marge gauche;
$margetop=5; // marge haut;
$margeright=5; // marge droite;
$border=1; // 1 -> bordure 0 -> pas de bordure";
$C1=0; // couleur texte R";
$C2=0; // couleur texte V";
$C3=0; // couleur texte B";
$size=10; //taille POLICE";
$height=4.6; // hauteur ligne tableau ";
$align='L';
// fond 2 couleurs
$fond=1; // 0- > FOND transparent 1 -> fond";
$C1fond1=234; // couleur fond R ";
$C2fond1=240; // couleur fond V ";
$C3fond1=245; // couleur fond B ";
$C1fond2=255; // couleur fond R";

```

(suite sur la page suivante)

(suite de la page précédente)

```

$C2fond2=255;// couleur fond V";
$C3fond2=255;// couleur fond B";
// spe openelec
$flagsessionliste=0;// 1 -> affichage session liste ou 0 -> pas d'affichage";
// titre
$bordertitre=0; // 1 -> bordure 0 -> pas de bordure";
$aligntitre='L'; // L,C,R";
$heighttitre=10;// hauteur ligne titre";
$grastitre='B';//\ $gras='B' -> BOLD OU \ $gras='';
$fondtitre=0; //0- > FOND transparent 1 -> fond";
$C1titrefond=181;// couleur fond R";
$C2titrefond=182;// couleur fond V";
$C3titrefond=188;// couleur fond B";
$C1titre=75;// couleur texte R";
$C2titre=79;// couleur texte V";
$C3titre=81;// couleur texte B";
$sizetitre=15;
// entete colonne
$flag_entete=1;//entete colonne : 0 -> non affichage , 1 -> affichage";
$fondentete=1;// 0- > FOND transparent 1 -> fond";
$heightentete=10;//hauteur ligne entete colonne";
$C1fondentete=210;// couleur fond R";
$C2fondentete=216;// couleur fond V";
$C3fondentete=249;// couleur fond B";
$C1entetetxt=0;// couleur texte R";
$C2entetetxt=0;// couleur texte V";
$C3entetetxt=0;// couleur texte B";
$C1border=159;// couleur texte R";
$C2border=160;// couleur texte V";
$C3border=167;// couleur texte B";
$bt=1;// border lere et derniere ligne du tableau par page->0 ou 1";

?>

```

3.3.8.6 gen/dyn/etat.inc.php

Ce fichier n'est plus utilisé par le générateur depuis la version 4.0 et la gestion des éditions en base de données.

3.3.8.7 gen/dyn/sousetat.inc.php

Ce fichier n'est plus utilisé par le générateur depuis la version 4.0 et la gestion des éditions en base de données.

3.3.8.8 Limites du générateur

Nous décrivons ici les limites du générateur qui pourront être réglées dans des versions ultérieures

Lorsqu'un formulaire principal (FP) contient un champ nommé « libelle » ainsi qu'un sous formulaire (FS) contenant lui-même un champ « libelle », cela pose plusieurs problèmes :

- pas w3c compliant,
- lors de l'exécution de javascript sur le sélecteur (id) du champ, en effet l'id du champ est le nom de la colonne correspondante dans la bdd.

La solution de contournement exemple avec autocomplete ayant le même nom en form et sous form

1- en modification désactiver les onglets : `$option_tab_disabled_on_edit=true`; la méthode autocomplete ne peut être active que dans formulaire

2- surcharge de la méthode autocomplete dans `obj/om_formulaire.class.php`

```
if($this->correct){
    $this->text($champ, $validation, $DEBUG = false);
}else{
    ...
}
```

si la validation est ok, on n'utilise plus la méthode autocomplete dans le formulaire de retour qui gêne l'utilisation de cette même méthode dans le sous formulaire

3.4 Intégration

Ce chapitre aborde les composants hors de l'environnement nécessaire au framework mais pouvant apporter des fonctionnalités supplémentaires

Il est proposé dans ce chapitre d'utiliser qgis serveur comme serveur wms.

3.4.1 Installation de qgis serveur sur linux debian et ubuntu

Commandes d'installation

```
apt-get install apache2 libapache2-mod-fcgid qgis-mapserver
a2enmod cgi
service apache2 restart
```

Test de bon fonctionnement

Exemples d'URLs pour tester le bon fonctionnement du serveur :

```
http://XX.XX.XX.XX/cgi-bin/qgis_mapserv.fcgi?SERVICE=WMS&VERSION=1.3.0&
↳REQUEST=GetCapabilities
```

Liens de références

```
http://adrien.caillot.free.fr/?p=9395
http://hub.qgis.org/projects/quantum-gis/wiki/QGIS_Server_Tutorial
http://io.gchatelier.fr/blog/installation-qgis-server-publication-dun-projet-qgis/
```

3.5 Pour aller plus loin...

Sommaire

- *Pour aller plus loin...*
- *Les scripts spécifiques de l'application*
 - *Introduction*
 - *Réaliser un script complémentaire*
 - *Exemple*

3.5.1 Les scripts spécifiques de l'application

3.5.1.1 Introduction

Les méthodes spécifiques à l'application sont dans `obj/utils.class.php` qui héritent de la class `om_application.class.php` d "openmairie

Vous pouvez surcharger les classes d'`om_application.class.php` dans `utils.class.php`

Exemple : surcharge de la méthode `login()` pour conserver le service d'un utilisateur en variable session dans open-Courrier.

Ces classes contiennent les méthodes utilisées par le framework mais qui peuvent vous aider à développer les scripts complémentaires de votre application.

Les scripts complémentaires sont mis en répertoire `app /` et peuvent être créer pour :

- faire un traitement (remise à 0 d'un registre, archivage, export ...)
- faire un sous programme spécifique appelé par un formulaire (`bible.php` dans openCourrier)
- faire une recherche avec un affichage particulier

Les scripts javascripts sont mis dans le fichier `app/js/script.js`

Les images spécifiques sont stockées dans `app/img`

3.5.1.2 Réaliser un script complémentaire

Il est proposé ici de vous montrer comment réaliser ce script complémentaire

Le script commence obligatoirement par un appel à la bibliothèque `utils.class.php` et la creation d un objet `$f` :

```
require_once "../obj/utils.class.php";
$f = new utils(NULL,
               "courrier",
               _("recherche"),
               "ico_recherche.png",
               "recherche");
```

Les parametres de l'objet sont les suivants :

- flag : si flag= Null affichage complete
- nonhtml : pas d affichage
- htmlonly : tout les elements externes html avec body vide
- right : droit géré en `om_droit` - vide ne verifie pas
- title : titre affiché
- icon : icone affiché
- help : aide affiché

`utils.class.php` fait la Verification si l utilisateur est authentifié et si l utilisateur a le droit (`util.class` surcharge `core/om_application.class.php` qui contient les scripts de base du framework)

Si le paramètre « right » est vide vous pouvez faire appel aux méthodes suivantes

```
isAccredited() // a le droit ou pas
isAuthenticated // si non authentifié, il est rejeté

$f->setRight($obj); // affecte un droit d acces
$f->isAuthorized(); //verification que l utilisateur accède
```

(suite sur la page suivante)

(suite de la page précédente)

```
// Affectation des variables en dehors du constructeur
$f->setTitle($ent);
$f->setIcon($ico);
$f->setHelp($obj);
$f->setFlag(NULL);

// affichage
$f->display();
```

Pour **executer une requête dans un fichier sql** vous devez stocker votre requête dans le répertoire `sql/type_de_sgbd/nom_de_requete.inc` afin de préserver la portabilité de vos travaux sur d'autres sgbd :

```
// appel au fichier requête
include ("../sql/".$f->phptype."/courrier_scr.inc");

// lancement de la requete sql_courrier et test erreur
$res=$f->db->query($sql_courrier);
$f->isDatabaseError($res);
```

Pour **parcourir les enregistrements** vous utilisez les méthodes dbpear suivantes :

```
// du debut à la fin de la requête
while ($row=& $res->fetchRow(DB_FETCHMODE_ASSOC)) {
    // j'affiche le champ courrier
    echo $row['courrier'];
}
```

Pour **ecrire dans la base** vous pouvez utiliser les méthodes insert ou update mais vous pouvez utiliser la méthode `autoexecute` spécifique à db pear :

requête sql

```
$sql = "INSERT INTO ... ";

$res2 = $f -> db -> query($sql);

$f->isDatabaseError($res2);
```

ou avec un tableau \$valF

```
$obj = table

$valF[$obj]=$f-> db -> nextId(DB_PREFIXE.$obj);

$res1= $f-> db -> autoExecute(DB_PREFIXE.$obj,$valF,DB_AUTOQUERY_INSERT);

$f->isDatabaseError($res1);
```

Vous pouvez faire une **Description du role de la page** de la manière suivante

```
$description = _("Cette page vous permet de .. ");

$f->displayDescription($description);
```

Un **message d erreur** s'affiche suivant :

`$class` : qui est la classe css qui s'affiche sur l'element et qui peut être

« error » : pour le message erreur

« valid » : pour le message de validation

le *code* est le suivant

```
$message = _("Mot de passe actuel incorrect");
$f->displayMessage($class, $message);
```

Pour afficher un **fieldset**, le code est le suivant

```
echo "<fieldset class=\"cadre ui-corner-all ui-widget-content\">\n";

echo "\t<legend class=\"ui-corner-all ui-widget-content ui-state-active\">";

echo _("Courrier")."</legend>";

...
echo "</fieldset>
```

il peut être par défaut *ouvert*

```
echo "<fieldset class= ... collapsible\">\n";
```

ou il peut être *fermé*

```
echo "<fieldset ... startClosed\">\n";
```

Vous pouvez faire **appel a des scripts js complementaires** en utilisant la méthode

```
$f->addHTMLHeadJs(array("../js/formulairedyn.js", "../js/onglet.js"));
```

Pour la **gestion des accents**, il est conseillé de ne pas mettre d accent dans le code (utf8 au lieu de latin1-iso8859-1) et de mettre les accents dans la traduction

Pour définir le chemin par défaut pour l'*** upload de fichier**, il faut utiliser la méthode

```
$path=$f->getPathFolderTrs()
```

3.5.1.3 Exemple

Il est proposé de prendre l'exemple du traitement de la remise du registre a 0 dans openCourrier

```
// ENTETE NORMALISEE

/**
 * Cette page permet de remettre a 0 le registre
 *
 * @package openmairie_exemple
 * @version SVN : $Id: xxxx.php 311 2010-12-06 11:43:36 xxxxx $
 */

// CREATION DE L' OBJET $f

require_once "../obj/utils.class.php";
$f = new utils(NULL, "traitement", _("remise a 0 du registre"), "ico_registre.png",
↪ "recherche");
```

(suite sur la page suivante)

(suite de la page précédente)

```

// get
if (isset ($_GET['validation'])) {
    $validation=$_GET['validation'];
}else{
    $validation=0;
}

/**
 * Description de la page
 */

$description = _("Cette page vous permet de remettre a 0 le numero de registre ".
                "Ce traitement est a faire en debut d annee.");
$f->displayDescription($description);

// TEST VALIDATION
// SI = 0 affichage du numero de registre
// SI = 1 mise à 0 du registre et affichage du résultat

if($validation==0){
    $validation=1;

    // REQUETE DU REGISTRE

    $sql= "select id from registre_seq" ;
    $res1=$f->db->getOne($sql);
    $f->isDatabaseError($res1);

    // AFFICHAGE DANS UN FIELDSET

    echo "<fieldset class=\"cadre ui-corner-all ui-widget-content\">\n";
    echo "\t<legend class=\"ui-corner-all ui-widget-content ui-state-active\">";
    echo _("Registre ")."</legend>";
    if ($res1!=0){
        echo "<br>"._("le dernier no du registre est")." : &nbsp;&nbsp;&nbsp;".$res1."&nbsp;&nbsp;&nbsp;";
        =>&nbsp;&nbsp;&nbsp;";
    }else{
        echo "<br>"._("vous avez deja fait une remise a 0")."<br>";
    }
    echo "<form method=\"POST\" action=\"num_registre.php?validation=".
    $validation.\" name=f1>";
    echo "</fieldset>";

    // BOUTON DE VALIDATION
    echo "\t<div class=\"formControls\">";
    echo "<input type='submit' value='"._("remise a 0 du registre").
    "&nbsp;&nbsp;&nbsp;' >";
    echo "</div>";
    echo "</form>";

}else { // validation=1

    // VALORISATION DE $valF

```

(suite sur la page suivante)

(suite de la page précédente)

```
$valF=array();
$valF['id']=0;

// REQUETE MISE A JOUR avec autoExecute
$res2= $f->db->autoExecute("registre_seq",$valF,DB_AUTOQUERY_UPDATE);
$f->isDatabaseError($res2);

// AFFICHAGE DU RESULTAT AVEC UN FIELDSET
echo "<fieldset class=\"cadre ui-corner-all ui-widget-content\">\n";
echo "\t<legend class=\"ui-corner-all ui-widget-content ui-state-active\">";
echo _("Registre ")."</legend>";
echo "<center><b>."_("remise a 0 du registre reussie")."</b></center>";
echo "</fieldset>";

} //validation
```

Notes

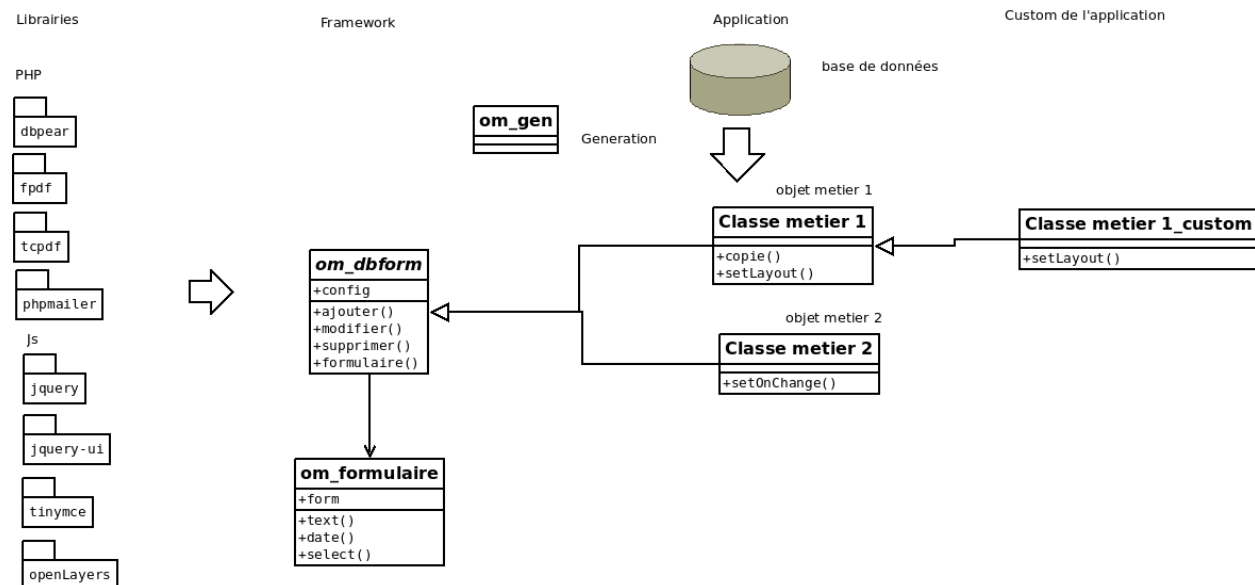
_(« Registre ») : _ (« texte ») permet l'utilisation de poedit pour la traduction de texte

class= »cadre ui-corner-all ui-widget-content » : suivant css de jquery

Il est proposé ici de décrire le fonctionnement du framework openMairie.

« En programmation orientée objet un framework est typiquement composé de classes mères qui seront dérivées et étendues par héritage en fonction des besoins spécifiques à chaque logiciel qui utilise le framework ».

<http://fr.wikipedia.org/wiki/Framework>



openMairie intègre de nombreux composants :

- FPDF : <http://fpdf.org/>
- TCPDF : <https://tcpdf.org/>
- DBPEAR : <http://pear.php.net/package/DB/redirected>
- PHPMailer : <https://github.com/PHPMailer/PHPMailer>
- JQUERY : <https://jquery.com/>
- JQUERY-UI : <http://jqueryui.com/>
- OPENLAYERS : <https://openlayers.org/>

— TINYMCE : <https://www.tinymce.com/>

Le framework est composé de 2 classes principales :

om_dbform est une classe abstraite surchargée par les objets métiers qui fait le lien entre la base de données et le formulaire

om_formulaire est la classe qui construit le formulaire en utilisant l'objet form.

Les classes métiers sont générées par le générateur une fois la base de données créée et surchargent la classe abstraite om_dbform.

Enfin, il est possible de surcharger les classes métiers par des classes customisées.

4.1 Arborescence

Sommaire

- *Arborescence*
 - *Introduction*
 - *Les répertoires spécifiques à l'applicatif*
 - *data/*
 - *app/*
 - *dyn/*
 - *gen/*
 - *locales/*
 - *obj/*
 - *sql/*
 - *tests/*
 - *var/*
 - *Les répertoires du framework*
 - *php/*
 - *core/*
 - *css/*
 - *img/*
 - *js/*
 - *lib/*
 - *pdf/*
 - *scr/*
 - *spg/*

4.1.1 Introduction

Cette rubrique vise à décrire brièvement l'arborescence du framework pour comprendre l'objectif de chaque répertoire. Elle est divisée en deux parties : les répertoires spécifiques à l'applicatif qui sont modifiés lors du développement de l'applicatif et les répertoires du framework qui sont récupérés tel quel dans le framework.

[F] `.htaccess` dans les descriptions de répertoires ci-dessous représente des fichiers `.htaccess` empêchant l'accès dans les répertoires en question qui ne doivent pas être accessibles depuis l'interface par l'utilisateur.

4.1.2 Les répertoires spécifiques à l'applicatif

Ces répertoires sont ceux qui font l'applicatif. Par exemple, sur un applicatif comme openCimetière ce sont les répertoires qui vont se trouver dans le gestionnaire de fichiers que l'on appelle les répertoires spécifiques.

4.1.2.1 data/

Contient tous les fichiers d'initialisation de la base de données de l'applicatif

```
[D] data/
|-[F] .htaccess
|-[D] pgsql/
|---- ...
|---- ...
```

4.1.2.2 app/

Contient tout les scripts spécifiques à l'applicatif que ce soit des javascripts ou des images ou des scripts PHP

```
[D] app/
|-[D] css/
|---- ...
|-[D] img/
|---- ...
|-[D] js/
|---- ...
|---- ...
```

4.1.2.3 dyn/

Contient les fichiers de paramétrage de l'applicatif

```
[D] dyn
|-[F] .htaccess
|---- ...
```

4.1.2.4 gen/

Contient les scripts (obj) et requêtes (sql) générés par le générateur

```
[D] gen
|-[F] .htaccess
|-[D] dyn/
|---- ...
|-[D] inc/
|---- ...
|-[D] obj/
|---- ...
|-[D] sql/
|-[D] pgsql
|---- ...
|---- ...
```

4.1.2.5 locales/

Contient les fichiers de traduction de l'applcatif

```
[D] locales/  
|- [F] .htaccess  
|---- ...
```

4.1.2.6 obj/

Contient les objets métiers surchargeant les objets générés

```
[D] obj/  
|- [F] .htaccess  
|---- ...
```

4.1.2.7 sql/

Contient les scripts sql surchargeant les scripts générés

```
[D] sql/  
|- [F] .htaccess  
|- [D] pgsql/  
|---- ...  
|---- ...
```

4.1.2.8 tests/

Contient les jeux de tests unitaires et fonctionnels de l'applcatif

```
[D] tests/  
|- [F] .htaccess  
|---- ...
```

4.1.2.9 var/

Contient les fichiers de logs, les fichiers temporaires et les fichiers stockés par l'applcatif

```
[D] var/  
|- [F] .htaccess  
|- [D] log/  
|   |- [F] error.log  
|   |---- ...  
|- [D] filestorage/  
|   |---- ...  
|- [D] tmp/  
|   |---- ...  
|---- ...
```

4.1.3 Les répertoires du framework

Ces répertoires sont ceux qui sont issus du framework, c'est-à-dire qu'ils ne sont pas dans l'applicatif lui même. Par exemple, sur un applicatif comme openCimetière ce sont les répertoires qui vont être récupérés par une propriété externes sur le gestionnaire de versions que l'on appelle les répertoires du framework.

4.1.3.1 php/

Contient les librairies PHP utilisées par le framework (comme dbpear, phpmailer ou fpdf)

```
[D] php
|-[F] .htaccess
|---- ...
```

4.1.3.2 core/

Contient les classes de la librairie du framework

```
[D] core
|-[F] .htaccess
|---- ...
```

4.1.3.3 css/

Contient les feuilles de style de base du framework

```
[D] css/
|---- ...
```

4.1.3.4 img/

Contient les images du framework

```
[D] img/
|---- ...
```

4.1.3.5 js/

Contient les javascripts de base du framework

```
[D] js/
|---- ...
```

4.1.3.6 lib/

Contient les librairies javascripts utilisées par le framework (comme openLayers ou jquery)

```
[D] lib/
|---- ...
```

4.1.3.7 pdf/

Contient les scripts d'édition du framework

```
[D] pdf/  
|---- ...
```

4.1.3.8 scr/

Contient les scripts d'affichage du framework

```
[D] scr/  
|---- ...
```

4.1.3.9 spg/

Contient les sous programmes génériques du framework

```
[D] spg/  
|---- ...
```

4.2 Initialisation de la base de données

Sommaire

- *Initialisation de la base de données*
- *Description du dossier data/pgsql/*
 - *Description de tous les fichiers init*.sql*
 - *Le fichier init.sql*
 - *Les fichiers init_metier*.sql*
 - *Les fichiers init_parametrage*.sql*
 - *Le fichier init_data.sql*
 - *Description des fichiers vX.X.X.sql ou ver_X.X.X.sql*
 - *Description du fichier update_sequences.sql*
 - *Description du fichier install.sql*
- *Paramétrage de la connexion à la base de données*

4.2.1 Description du dossier data/pgsql/

Il est nécessaire de positionner l'entête de fichier suivant pour chacun des fichiers sql de ce dossier :

```
-----  
-- Description succincte de l'utilité du fichier  
--  
-- Informations nécessaires à la génération ou à la composition du fichier  
--  
-- @package <APPLICATIF>
```

(suite sur la page suivante)

(suite de la page précédente)

```
-- @version SVN : $Id$
```

4.2.1.1 Description de tous les fichiers `init*.sql`

Il s'avère nécessaire de mettre dans l'entête des fichiers `init*.sql` la commande ou les instructions qui ont permis de générer ou de composer le fichier en question.

4.2.1.1.1 Le fichier `init.sql`

Ce fichier contient les instructions de base du framework. Il permet de créer les tables et les séquences du framework (celles qui commencent par `om_*`). Il est généré grâce à la commande :

```
pg_dump -s -O -n <SCHEMA> -T <SCHEMA>.om_* <DATABASE>
```

Dans les applicatifs, ce fichier est sensé être directement copié depuis le framework.

4.2.1.1.2 Les fichiers `init_metier*.sql`

Ces fichiers contiennent les instructions de base de l'applicatif.

- Le fichier `init_metier.sql` permet de modifier (si besoin) le modèle de données du framework et de créer les tables et les séquences de l'applicatif (celles qui ne commencent par `om_*`). Il est généré grâce à la commande :

```
pg_dump -s -O -n <SCHEMA> -t <SCHEMA>.om_* <DATABASE>
```

Dans le framework, ce fichier est vide.

- Le fichier `init_metier_sig.sql` permet de modifier le modèle de données créé précédemment pour y ajouter des champs de type `geom` pour la gestion du SIG.
- Le fichier `init_metier_vue.sql` permet de modifier le modèle de données créé précédemment pour y remplacer une table par une vue vers la table d'un autre schéma ou d'une autre base de données.

4.2.1.1.3 Les fichiers `init_parametrage*.sql`

Ces fichiers contiennent l'initialisation du paramétrage c'est-à-dire les données nécessaires à l'utilisation de l'application. Ils sont générés généralement grâce à la commande :

```
pg_dump -a -t <SCHEMA>.<TABLE1> -t <SCHEMA>.<TABLE2> ... <DATABASE>
```

- Le fichier `init_parametrage.sql` permet d'initialiser par exemple la ou les collectivités de base ainsi que les profils et l'utilisateur admin. Dans certains applicatifs simple, ce fichier peut être unique et tout le paramétrage contenu dans ce dernier.
- Le fichier `init_parametrage_permissions.sql` permet d'initialiser les permissions de l'applicatif. Cette initialisation se trouve dans un fichier séparé pour appréhender plus facilement le paramétrage des permissions et éventuellement la mise à jour de ce paramétrage.
- Le fichier `init_parametrage_editions.sql` permet d'initialiser les éditions de base générique de l'applicatif. Cette initialisation se trouve dans un fichier séparé pour appréhender plus facilement le paramétrage des éditions et éventuellement la mise à jour de ce paramétrage.
- Le fichier `init_parametrage_*.sql` peut permettre de découper encore l'initialisation pour appréhender plus facilement le paramétrage et éventuellement la mise à jour de ce paramétrage.

4.2.1.1.4 Le fichier `init_data.sql`

Ce fichier contient l'initialisation d'un jeu de données à destination de trois environnements distincts :

- l'environnement de développement,
- l'environnement de démonstration,
- l'environnement de tests.

4.2.1.2 Description des fichiers `vX.X.X.sql` ou `ver_X.X.X.sql`

Ces fichiers permettent de mettre à jour les applicatifs d'une version vers la version supérieure. Le X.X.X correspond au numéro de version vers lequel la mise à jour se fait et depuis la version juste précédente.

Lorsque le framework ou l'applicatif est en développement, ce fichier peut être suffixé par `-dev` et indique qu'il n'a pas encore été intégré aux différents fichiers `init*.sql`. Juste avant une nouvelle version du framework, les fichiers `init*.sql` doivent être régénérés pour intégrer les dernières modifications et ce fichier renommé avec son nom `vX.X.X.sql` ou `ver_X.X.X.sql` standard.

4.2.1.3 Description du fichier `update_sequences.sql`

Ce fichier permet de créer une fonction capable de mettre à jour toutes les séquences correctement liées aux champs auxquels elles se rattachent en fonction de la dernière valeur du champ dans la table. En plus de la création de la fonction ce script exécute la fonction.

4.2.1.4 Description du fichier `install.sql`

Ce fichier permet d'initialiser tous les fichiers qui sont décrits ci-dessus dans le bon ordre. Par défaut ce fichier installe la base de données et les données nécessaires aux trois environnements suivants :

- l'environnement de développement,
- l'environnement de démonstration,
- l'environnement de tests.

Note : Ce fichier comporte l'initialisation des commandes postgres par défaut pour la dernière version de postgres. Les commandes pour l'ancienne version sont présentes et commentées dans ce même fichier.

4.2.2 Paramétrage de la connexion à la base de données

Le paramétrage de la connexion à la base de données se fait dans le fichier `dyn/database.inc.php`.

Note : Dans le framework le schéma utilisé par défaut est *openexemple*, dans les applicatifs c'est normalement le nom de l'applicatif *<APPLICATIF>* (par exemple : *openelec*, *opencimetiere*, ...).

```
<?php
/**
 * Ce fichier permet le paramétrage de la connexion à la base de données,
 * chaque entrée du tableau correspond à une base différente. Attention
 * l'index du tableau conn représente l'identifiant du dossier dans lequel
 * seront stockés les fichiers propres à cette base dans l'application.
 *
 * @package openmairie_exemple
```

(suite sur la page suivante)

(suite de la page précédente)

```

* @version SVN : $Id: database.inc.php 2302 2013-05-23 18:04:22Z fmichon $
*/

// PostgreSQL
$conn[1] = array(
    "openExemple", // Titre
    "pgsql", // Type de base
    "pgsql", // Type de base
    "postgres", // Login
    "postgres", // Mot de passe
    "tcp", // Protocole de connexion
    "localhost", // Nom d'hôte
    "5432", // Port du serveur
    "", // Socket
    "openexemple", // Nom de la base
    "AAAA-MM-JJ", // Format de la date
    "openexemple", // Nom du schéma
    "", // Préfixe
    NULL, // Paramétrage pour l'annuaire LDAP
    "mail-default", // Paramétrage pour le serveur de mail
    "filestorage-default", // Paramétrage pour le stockage des fichiers
    "extras" => array( // Paramétrage optionnel, utilisé pour les plugins.
        "sig" => "",
    ),
);

?>

```

L'attribut optionnel “extras” est un tableau associatif qui contient un ou plusieurs paramétrage de base de données. La méthode `get_database_extra_parameters` de la classe `om_application` permet de récupérer ces paramètres.

La documentation de DB PEAR qui est le module d'abstraction utilisé par le framework donne plus d'informations sur les paramètres.

4.3 Paramétrage du framework

Sommaire

- *Paramétrage du framework*
 - *Introduction*
 - *Les scripts de paramétrage*
 - *Le serveur d'envoi de mail*
 - *L'annuaire LDAP*
 - *Les zones de navigation*
 - *Le menu*
 - *Les actions personnelles*
 - *Les raccourcis*
 - *Les actions globales*
 - *Les variables locales et la langue*
 - *Le paramétrage de l'application métier*
 - *Le nom de l'application*
 - *Le titre HTML de l'application*

- *Le nom de la session*
- *Le mode démonstration*
- *La redéfinition du mot de passe oublié par l'utilisateur*
- *Le nombre de colonnes du tableau de bord*
- *Le favicon de l'application*
- *Le mode de gestion des permissions*
- *La valeur par défaut lorsqu'une permission n'existe pas*
- *Les extensions de fichiers autorisées*
- *La taille maximale de fichiers autorisée*
- *Le Paramétrage des librairies*
- *Le mode DEBUG*
- *La version de votre application*
- *Les informations générales*
- *L'installation automatique*
- *Les paramètres des combos*
- *Les paramètres éditions*
- *Les paramètres om_sig*

4.3.1 Introduction

Le paramétrage de l'application se fait dans le répertoire `dyn/`. Il est proposé dans ce chapitre de décrire les différents scripts de paramétrage et leur utilisation.

4.3.2 Les scripts de paramétrage

- `dyn/database.inc.php` : voir *Paramétrage de la connexion à la base de données*.
- `dyn/config.inc.php` : application
- `dyn/filestorage.inc.php` : système de stockage des fichiers
- `dyn/mail.inc.php` : serveur smtp pour l'envoi de mails
- `dyn/directory.inc.php` : annuaire LDAP pour la synchronisation des utilisateurs
- `dyn/custom.inc.php` : ...
- `dyn/menu.inc.php` : zone de navigation "Menu"
- `dyn/actions.inc.php` : zone de navigation "Actions personnelles"
- `dyn/shortslinks.inc.php` : zone de navigation "Raccourcis"
- `dyn/footer.inc.php` : zone de navigation "Actions globales"
- `dyn/locales.inc.php` : application
- `dyn/include.inc.php` : chemin d'accès aux librairies
- `dyn/debug.inc.php` : mode debug
- `dyn/version.inc.php` : paramétrage de la version
- `dyn/comboaffichage.inc.php` : paramétrage combo
- `dyn/comboparametre.inc.php` : paramétrage combo
- `dyn/comboretour.inc.php` : paramétrage combo
- `dyn/tab.inc.php` : variable spécifique à passer dans l'url pour `tab.php`
- `dyn/soustab.php` : variable spécifique à passer dans l'url pour `soustab.php`
- `dyn/form.inc.php` : variable spécifique à passer dans l'url pour `form.php`
- `dyn/sousform.inc.php` : variable spécifique à passer dans l'url pour `sous.php`
- `dyn/form.get.specific.inc.php`
- `dyn/sousform.get.specific.inc.php` : variable spécifique sousform
- `dyn/varetatpdf.inc` : variable état et sousetat pdf
- `dyn/varlettretypedpdf.inc` : variable lettre type
- `dyn/varsousetatpdf.inc` : ...

- `dyn/var.inc` : variable application (deprecated : préférez `om_parametre`)
- `dyn/var_sig.inc` : paramétrage sig
- `dyn/var_adresse_postale.inc` : paramétrage sig

4.3.3 Le serveur d'envoi de mail

Niveau de configuration “INSTANCE”.

Le script `dyn/mail.inc.php` permet de configurer le serveur SMTP qui va être utilisé pour envoyer tous les mails depuis l'application. L'unique fonctionnalité dans le framework qui gère un envoi de mail est la réinitialisation du mot de passe oublié par un utilisateur.

La variable `$mail` est un tableau associatif. Ce tableau peut, de ce fait, contenir plusieurs configurations de serveur mail différentes. Chaque serveur est représenté par une clé de tableau. Ces clés se retrouvent dans le script `dyn/database.inc.php` et permettent d'associer une base de données précise à un serveur mail précis.

Les clés de configuration sont :

- `mail_host` -> Adresse du serveur de mail
- `mail_port` -> Port d'écoute du serveur de mail
- `mail_username` -> Identifiant de l'utilisateur du serveur de mail
- `mail_pass` -> Mot de passe de cet utilisateur
- `mail_from` -> Adresse email de l'expéditeur
- `mail_from_name` -> Nom de l'expéditeur

Exemple d'un fichier de configuration :

```
<?php
$mail = array();
$mail["mail-default"] = array(
    'mail_host' => 'mail.exemple.com',
    'mail_port' => '25',
    'mail_username' => 'nospam@exemple.com',
    'mail_pass' => 'mot_de_passe',
    'mail_from' => 'contact@exemple.com',
    'mail_from_name' => 'exemple',
);
?>
```

4.3.4 L'annuaire LDAP

Niveau de configuration “INSTANCE”.

Le script `dyn/directory.inc.php` permet de configurer le serveur LDAP qui va être utilisé pour synchroniser les utilisateurs et les authentifier.

La variable `$directory` est un tableau associatif. Ce tableau peut, de ce fait, contenir plusieurs configurations d'annuaires LDAP différentes. Chaque connexion est représentée par une clé de tableau. Ces clés se retrouvent dans le script `dyn/database.inc.php` et permettent d'associer une base de données précise à un annuaire LDAP précis.

Les clés de configuration sont :

- `ldap_server` -> Adresse du serveur LDAP
- `ldap_server_port` -> Port d'écoute du serveur LDAP
- `ldap_admin_login` -> identifiant de l'administrateur LDAP
- `ldap_admin_passwd` -> mot de passe de cet administrateur
- `ldap_base` -> Base de l'arbre LDAP
- `ldap_base_users` -> Base utilisateurs de l'arbre LDAP

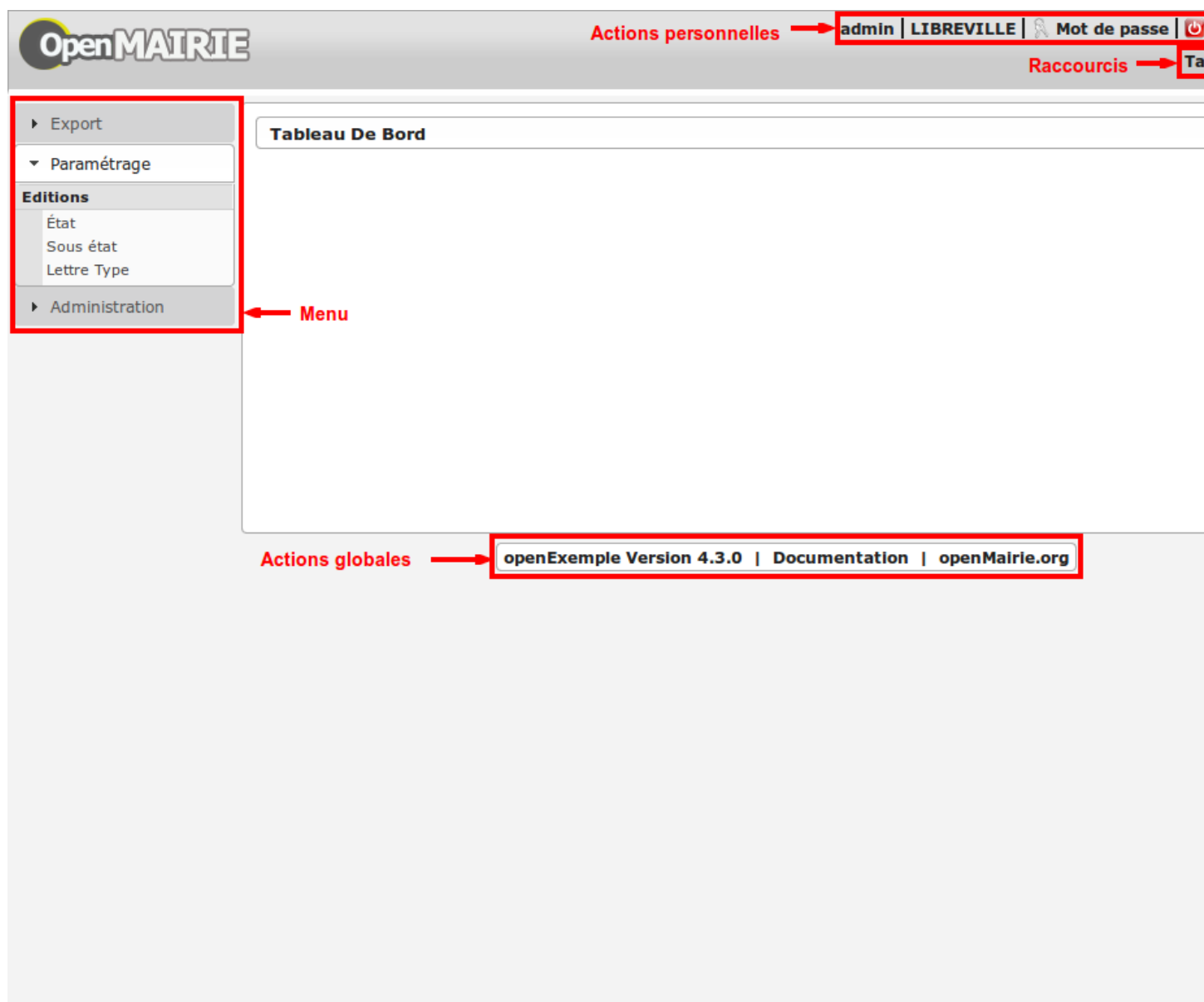
- ldap_user_filter -> Filtre utiliser par la fonction ldap_search
- ldap_login_attrib -> Attribut LDAP qui sera utilise comme login dans la base
- ldap_more_attrib -> Correspondance des champs entre l’annuaire et la base (Par exemple si on prend l’exemple de configuration ci dessous, la colonne “nom” de la base de données sera synchronisée avec l’attribut “name” de l’annuaire. De plus la colonne “email” sera synchronisée avec l’attribut “mail” de l’annuaire. Si l’attribut “mail” n’est pas trouvé dans le schéma LDAP, l’attribut “mailAddress” sera utilisé à la place. Il est possible de spécifier plusieurs attributs en utilisant un tableau de cette manière.)
- default_om_profil -> Profil des utilisateurs ajoutes depuis l’annuaire

Exemple d’un fichier de configuration :

```
<?php
$directory = array();
$directory["ldap-default"] = array(
    'ldap_server' => 'localhost',
    'ldap_server_port' => '389',
    'ldap_admin_login' => 'cn=admin,dc=openmairie,dc=org',
    'ldap_admin_passwd' => 'admin',
    'ldap_base' => 'dc=openmairie,dc=org',
    'ldap_base_users' => 'dc=openmairie,dc=org',
    'ldap_user_filter' => 'objectclass=person',
    'ldap_login_attrib' => 'cn',
    'ldap_more_attrib' => array(
        'nom' => 'name',
        'email' => array('mail', 'mailAddress'),
    ),
    'default_om_profil' => 1,
);
?>
```

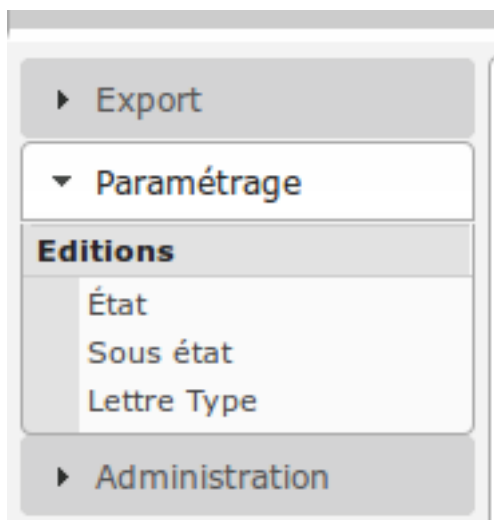
4.3.5 Les zones de navigation

Quatre zones de navigation différentes sont disponibles dans le framework :



4.3.5.1 Le menu

Le but de cette zone de navigation est de rassembler les liens vers toutes les fonctions du logiciel regroupées par rubrique et catégorie. Elle se situe à gauche du contenu et est visible uniquement lorsque l'utilisateur est authentifié.



Par défaut **le menu** est composé de la manière suivante

application	vide par défaut, contient l'accès à votre application
export	contient le script "edition" qui reprend les éditions pdf des tables contient le menu "reqmo" qui reprend les requêtes mémoires
traitement	vide par défaut, cet option contient les scripts de traitements
parametrage	Cette option contient vos tables de paramétrage fonctionnel. Par défaut il contient le paramétrage des états / sous-états / lettres type
administration	Cette option contient les fonctions de configuration de l'administrateur technique. Cela comprend notamment le paramétrage de la collectivité, om_sig et la gestion des droits d'accès

La configuration des liens se fait dans le fichier `dyn/menu.inc.php`. Ce fichier de paramétrage n'est pas obligatoire. Si il n'existe pas, aucun lien n'est affiché. Ce fichier de paramétrage doit contenir la déclaration d'un tableau de tableaux associatifs dans la variable `$menu`. Chaque tableau associatif représente une rubrique. Chaque rubrique contient un tableau de tableaux associatifs, chacun représentant un lien.

Les caractéristiques de ce tableau sont les suivantes :

tableau rubrik

```
title (obligatoire)
description (texte qui s'affiche au survol de la rubrique)
href (contenu du lien href)
class (classe css qui s'affiche sur la rubrique)
right (droit que l'utilisateur doit avoir pour visionner cette rubrique)
links (obligatoire)
open (critères de pré-ouverture de cette rubrique du menu)
```

tableau links

```
title (obligatoire)
href (obligatoire) (contenu du lien href)
class (classe css qui s'affiche sur l'element)
right (droit que l'utilisateur doit avoir pour visionner cet element)
target (pour ouvrir le lien dans une nouvelle fenetre)
```

(suite sur la page suivante)

(suite de la page précédente)

```
open (critères de pré-ouverture de la rubrique du menu dans laquelle est ce
    lien, et sélection de ce lien en lien actif)
// condition d'affichage de l'option suivant valeur d'un om parametre
"parameters" => array("option_courrier_depart" => 'true', ),
```

L'entrée `open` sert à marquer une entrée de menu comme active. La rubrique contenant cette entrée est ouverte dès l'affichage de la page, et l'entrée active est mise en évidence. L'entrée `open` peut contenir soit une chaîne soit un `array()` comportant plusieurs chaînes. Chaque chaîne est créée selon la syntaxe `'script.php|obj'`, chacune des deux parties étant optionnelle. Le caractère séparateur `|` est obligatoire.

Exemple `['\|om_collectivite'` sélectionnera l'entrée pour toutes les url] ayant `obj=om_collectivite`
`'tab.php|om_collectivite'` sélectionnera l'entrée pour l'affichage du tableau de la classe `om_collectivite`
`'unecran.php|'` sélectionnera l'entrée dès lors que le script `unecran.php` est appelé quelque soit la classe `obj`

4.3.5.2 Les actions personnelles

Le but de cette zone de navigation est de regrouper des liens vers des fonctions qui concernent les informations de connexion de l'utilisateur. Elle se situe dans le coin en haut à droite de l'écran et est visible uniquement lorsque l'utilisateur est authentifié.



Par défaut **les actions personnelles** sont composées de quatre éléments :

- le login de l'utilisateur,
- le libellé de la collectivité,
- un lien vers la page de modification du mot de passe,
- un lien vers la page de déconnexion du logiciel.

Le login de l'utilisateur est récupéré par la méthode `displayActionLogin()` de la classe `om_application`. Cette méthode peut être surchargée dans la classe `utils`.

Le libellé de la collectivité est récupéré par la méthode `displayActionCollectivite()` de la classe `om_application`. Cette méthode peut être surchargée dans la classe `utils`.

La configuration des liens se fait dans le fichier `dyn/actions.inc.php`. Ce fichier de paramétrage n'est pas obligatoire. Si il n'existe pas, aucun lien n'est affiché. Ce fichier de paramétrage doit contenir la déclaration d'un tableau de tableaux associatifs dans la variable `$actions`. Chaque tableau associatif représente un lien.

```
<?php
//
$actions = array();
//
$actions[] = array(
    "title" => _("Link"),
    "description" => _("Description"),
    "href" => "../scr/link.php",
    "target" => "_blank",
```

(suite sur la page suivante)

(suite de la page précédente)

```

    "class" => "action-link",
    "right" => "link",
);
?>

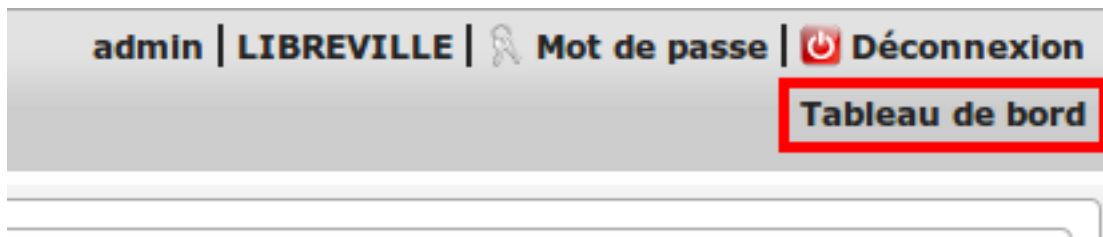
```

Description de chaque paramètre du tableau associatif :

Paramètre	Requis ?	Description
title	O	Texte
description	N	Texte qui s'affiche au survol de l'élément
href	N	Contenu du lien href
target	N	Attribut pour ouvrir le lien dans une nouvelle fenêtre
class	N	Classe CSS qui s'affiche sur l'élément
right	N	Permission nécessaire à l'utilisateur pour visualiser l'élément

4.3.5.3 Les raccourcis

Le but de cette zone de navigation est de regrouper des liens vers des fonctions précises utilisées très souvent. Elle se situe en haut à droite de l'écran juste au dessous des actions personnelles et est visible uniquement lorsque l'utilisateur est authentifié.



Par défaut **les raccourcis** contiennent uniquement un lien vers le tableau de bord.

La configuration des liens se fait dans le fichier `dyn/shortlinks.inc.php`. Ce fichier de paramétrage n'est pas obligatoire. Si il n'existe pas, aucun lien n'est affiché. Ce fichier de paramétrage doit contenir la déclaration d'un tableau de tableaux associatifs dans la variable `$shortlinks`. Chaque tableau associatif représente un lien.

```

<?php
// On initialise le tableau conteneur
$shortlinks = array();
// On ajoute au tableau conteneur un tableau associatif représentant un lien
// (à répéter autant de fois que nécessaire)
$shortlinks[] = array(
    "title" => _("Link"),
    "description" => _("Description"),
    "href" => "../scr/link.php",
    "target" => "_blank",
    "class" => "action-link",
    "right" => "link",
);
?>

```

Paramètre	Requis ?	Description
title	O	Texte
description	N	Texte qui s'affiche au survol de l'élément
href	N	Contenu du lien href
target	N	Attribut pour ouvrir le lien dans une nouvelle fenêtre
class	N	Classe CSS qui s'affiche sur l'élément
right	N	Permission nécessaire à l'utilisateur pour visualiser l'élément

4.3.5.4 Les actions globales

Le but de cette zone de navigation est de représenter la section « À propos » du logiciel. Elle se situe en bas de l'écran juste au dessous du contenu de la page et est visible lorsque l'utilisateur est authentifié ou non.

openExemple Version 4.3.0 | Documentation | openMairie.org

Par défaut **les actions globales** sont composées de trois éléments :

- le nom du logiciel ainsi que son numéro de version,
- un lien vers la documentation du site openMairie,
- un lien vers le site openMairie.

Le nom du logiciel est récupéré de la variable `$config['application']` présente dans le fichier `dyn/config.inc.php`. La version est récupérée de la variable `$version` présente dans le fichier `dyn/version.inc.php`.

La configuration des liens se fait dans le fichier `dyn/footer.inc.php`. Ce fichier de paramétrage n'est pas obligatoire. Si il n'existe pas, aucun lien n'est affiché. Ce fichier de paramétrage doit contenir la déclaration d'un tableau de tableaux associatifs dans la variable `$footer`. Chaque tableau associatif représente un lien.

```
<?php
// On initialise le tableau conteneur
$footer = array();
// On ajoute au tableau conteneur un tableau associatif représentant un lien
// (à répéter autant de fois que nécessaire)
$footer[] = array(
    "title" => _("Link"),
    "description" => _("Description"),
    "href" => "../scr/link.php",
    "target" => "_blank",
    "class" => "action-link",
    "right" => "link",
);
?>
```

Paramètre	Requis ?	Description
title	O	Texte
description	N	Texte qui s'affiche au survol de l'élément
href	N	Contenu du lien href
target	N	Attribut pour ouvrir le lien dans une nouvelle fenêtre
class	N	Classe CSS qui s'affiche sur l'élément
right	N	Permission nécessaire à l'utilisateur pour visualiser l'élément

4.3.6 Les variables locales et la langue

Les variables locales sont paramétrées dans le fichier *dyn/locales.inc.php*

Ce fichier contient :

- le paramétrage du codage des caracteres (ISO-8859-1 ou UTF8)

```
"DEPRECATED"

define('CHARSET', 'ISO-8859-1');
ou
define('CHARSET', 'UTF8');

Dans la version 4.2.0, il y a 2 paramètres :

pour la base : DB_CHARSET
pour apache : HTTP_CHARSET

Ces 2 paramètres remplacent CHARSET

Note ::

Dans apache, il est possible de modifier l'encodage
dans etc/apache2/apache2.conf commenter ##AddDefaultCharset = ISO-8859-1
relancer ensuite apache : $ etc/apache2/init.d/apache2 reload

A partir de la version 3.0.1, l'incompatibilité utf8 de la bibliothèque fpdf
→ est traitée
```

- le dossier où sont installées les variables du système

```
define('LOCALE', 'fr_FR');
```

- Le dossier contenant les locales et les fichiers de traduction

```
define('LOCALES_DIRECTORY', '../locales');
```

- Le domaine de traduction

```
define('DOMAIN', 'openmairie');
```

Les zones à traduire sont sous le format : _ (« zone à traduire »)

Voir le chapitre sur les outils : *poEdit*

4.3.7 Le paramétrage de l'application métier

L'application métier est paramétrée dans *dyn/var.inc*

Ce script contient les paramètres globaux de l'application. Attention les paramètres s'appliquent à toutes les bases de l'application.

Le paramétrage spécifique par collectivité doit se faire dans la table *om_parametre*

La configuration générale de l'application se fait aussi dans *dyn/config.inc.php*.

Les paramètres sont récupérés avec la création d'un objet utils par : `$f->config["nom_du_parametre"]`

4.3.7.1 Le nom de l'application

C'est le nom de l'application, il est utilisé pour l'affichage dans le footer (juste avant les actions globales) et dans le générateur pour distinguer la génération des mots clés du core et ceux de l'application.

Trois niveaux de configuration sont disponibles pour cet élément : framework, application et instance. Voici l'ordre de préférence si les trois niveaux sont configurés : instance > application > framework.

Pour configurer au niveau de l'instance, il faut définir dans le script `dyn/config.inc.php` le paramètre **application** sur le tableau `$config`.

```
<?php
$config = array();
$config["application"] = "openExemple";
?>
```

Pour configurer au niveau de l'application, il faut définir dans la classe `utils` définie dans le script `obj/utils.class.php` l'attribut `$_application_name`.

```
<?php
...
class utils extends application {

    /**
     * Gestion du nom de l'application.
     *
     * @var mixed Configuration niveau application.
     */
    protected $_application_name = "openExemple";
    ...
?>
```

Une configuration par défaut est définie dans le framework, dans la classe `application` définie dans le script `core/om_application.class.php` l'attribut `$_application_name`.

```
<?php
...
class application {

    /**
     * Gestion du nom de l'application.
     *
     * @var mixed Configuration niveau framework.
     */
    protected $_application_name = "openMairie";
    ...
?>
```

Pour récupérer la valeur du paramètre sans se préoccuper d'où vient le paramètre l'accessor `application::get_config__html_head_title()` est disponible. C'est toujours cette méthode qui doit être utilisée pour accéder au paramètre. Exemple d'utilisation :

```
<?php
...
$f->get_config__html_head_title();
...
?>
```

4.3.7.2 Le titre HTML de l'application

C'est le contenu de l'attribut titre de la page HTML, il est utilisé dans :

- le titre de l'onglet du navigateur,
- le titre du favori lorsque la page y est ajouté.

Trois niveaux de configuration sont disponibles pour cet élément : framework, application et instance. Voici l'ordre de préférence si les trois niveaux sont configurés : instance > application > framework.

Pour configurer au niveau de l'instance, il faut définir dans le script `dyn/config.inc.php` le paramètre **title** sur le tableau `$config`.

```
<?php
$config = array();
$config["title"] = ":: openMairie :: openExemple - Framework";
?>
```

Pour configurer au niveau de l'application, il faut définir dans la classe `utils` définie dans le script `obj/utils.class.php` l'attribut `$html_head_title`.

```
<?php
...
class utils extends application {

    /**
     * Titre HTML.
     *
     * @var mixed Configuration niveau application.
     */
    protected $html_head_title = ":: openMairie :: openExemple - Framework";
    ...
?>
```

Une configuration par défaut est définie dans le framework, dans la classe `application` définie dans le script `core/om_application.class.php` l'attribut `$html_head_title`.

```
<?php
...
class application {

    /**
     * Titre HTML.
     *
     * @var mixed Configuration niveau framework.
     */
    var $html_head_title = ":: openMairie ::";
    ...
?>
```

Pour récupérer la valeur du paramètre sans se préoccuper d'où vient le paramètre l'accesseur `application::get_config_html_head_title()` est disponible. C'est toujours cette méthode qui doit être utilisée pour accéder au paramètre. Exemple d'utilisation :

```
<?php
...
$f->get_config_html_head_title();
...
?>
```

4.3.7.3 Le nom de la session

Ce paramètre permet de spécifier le nom de la session. Il est important que chaque instance d'application possède un nom de session différent afin d'éviter des conflits de connexion entre plusieurs instances. Le nom de session est utilisé comme nom pour les cookies et les URLs (i.e. PHPSESSID). Il ne doit contenir que des caractères alphanumériques ; il doit être court et descriptif (surtout pour les utilisateurs ayant activé l'alerte cookie). Voir : <http://php.net/manual/fr/function.session-name.php>.

Trois niveaux de configuration sont disponibles pour cet élément : framework, application et instance. Voici l'ordre de préférence si les trois niveaux sont configurés : instance > application > framework.

Pour configurer au niveau de l'instance, il faut définir dans le script `dyn/config.inc.php` le paramètre `session_name` sur le tableau `$config`.

```
<?php
$config = array();
$config["session_name"] = "a2f587f1425bba47a8";
?>
```

Pour configurer au niveau de l'application, il faut définir dans la classe `utils` définie dans le script `obj/utils.class.php` l'attribut `$_session_name`.

```
<?php
...
class utils extends application {

    /**
     * Gestion du nom de la session.
     *
     * @var mixed Configuration niveau application.
     */
    var $_session_name = "c3f587f1425bba47a8";
    ...
?>
```

Une configuration par défaut est définie dans le framework, dans la classe `application` définie dans le script `core/om_application.class.php` l'attribut `$_session_name`.

```
<?php
...
class application {

    /**
     * Gestion du nom de la session.
     *
     * @var mixed Configuration niveau framework.
     */
    protected $_session_name = "1bb484de79f96a7d0b00ff463c18fcbf";
    ...
?>
```

Pour récupérer la valeur du paramètre sans se préoccuper d'où vient le paramètre l'accessoire `application::get_session_name()` est disponible. C'est toujours cette méthode qui doit être utilisée pour accéder au paramètre. Exemple d'utilisation :

```
<?php
...
```

(suite sur la page suivante)

(suite de la page précédente)

```
$f->get_session_name();  
...  
?>
```

4.3.7.4 Le mode démonstration

Ce paramètre permet de spécifier si l'instance de l'application se trouve en mode démonstration ou non. Ce mode permet de pré-remplir le formulaire de login avec l'identifiant "demo" et le mot de passe "demo". Par défaut, ce paramètre est positionné à "false" et peut donc éventuellement être surchargé au niveau de l'instance. Il suffit de définir dans le script `dyn/config.inc.php` le paramètre **demo** sur le tableau `$config`. Important : Pour empêcher l'utilisateur ainsi connecter de changer le mot de passe, il faut supprimer la permission au profil de l'utilisateur.

```
<?php  
$config = array();  
$config["demo"] = true;  
?>
```

4.3.7.5 La redéfinition du mot de passe oublié par l'utilisateur

Ce paramètre permet d'activer ou non la redéfinition de son mot de passe en cas d'oubli via un lien sur le formulaire de login. Par défaut, ce paramètre est positionné à "false" et peut donc éventuellement être surchargé au niveau de l'instance. Il suffit de définir dans le script `dyn/config.inc.php` le paramètre **password_reset** sur le tableau `$config`. Important : La réinitialisation du mot de passe est effectuée par un envoi de mail, il est donc nécessaire d'avoir configuré un serveur mail au préalable.

```
<?php  
$config = array();  
$config["password_reset"] = true;  
?>
```

4.3.7.6 Le nombre de colonnes du tableau de bord

Ce paramètre permet de spécifier le nombre de colonnes présentes sur le tableau de bord de l'application. Important : la modification de ce paramètre doit être suivie de la modification des données dans la base car des widgets existent peut être dans des colonnes supprimées.

Trois niveaux de configuration sont disponibles pour cet élément : framework, application et instance. Voici l'ordre de préférence si les trois niveaux sont configurés : instance > application > framework.

Pour configurer au niveau de l'instance, il faut définir dans le script `dyn/config.inc.php` le paramètre **dashboard_nb_column** sur le tableau `$config`.

```
<?php  
$config = array();  
$config["dashboard_nb_column"] = 4;  
?>
```

Pour configurer au niveau de l'application, il faut définir dans la classe `utils` définie dans le script `obj/utils.class.php` l'attribut `$config__dashboard_nb_column`.

```
<?php
...
class utils extends application {

    /**
     * Gestion du nombre de colonnes du tableau de bord.
     *
     * @var mixed Configuration niveau application.
     */
    var $config__dashboard_nb_column = 2;
...
?>
```

Une configuration par défaut est définie dans le framework, dans la classe application définie dans le script `core/om_application.class.php` l'attribut `$config__dashboard_nb_column`.

```
<?php
...
class application {

    /**
     * Gestion du nombre de colonnes du tableau de bord.
     *
     * @var mixed Configuration niveau framework.
     */
    var $config__dashboard_nb_column = 3;
...
?>
```

Pour récupérer la valeur du paramètre sans se préoccuper d'où vient le paramètre l'accesseur `application::get_config__dashboard_nb_column()` est disponible. C'est toujours cette méthode qui doit être utilisée pour accéder au paramètre. Exemple d'utilisation :

```
<?php
...
$f->get_config__dashboard_nb_column();
...
?>
```

4.3.7.7 Le favicon de l'application

Ce paramètre permet de spécifier l'image utilisée comme favicon de l'application.

Trois niveaux de configuration sont disponibles pour cet élément : framework, application et instance. Voici l'ordre de préférence si les trois niveaux sont configurés : instance > application > framework.

Pour configurer au niveau de l'instance, il faut définir dans le script `dyn/config.inc.php` le paramètre **favicon** sur le tableau `$config`.

```
<?php
$config = array();
$config["favicon"] = "../custom/favicon.ico";
?>
```

Pour configurer au niveau de l'application, il faut définir dans la classe `utils` définie dans le script `obj/utils.class.php` l'attribut `$html_head_favicon`.

```
<?php
...
class utils extends application {

    /**
     * Gestion du favicon de l'application.
     *
     * @var mixed Configuration niveau application.
     */
    var $html_head_favicon = "../app/img/favicon.ico";
...
?>
```

Une configuration par défaut est définie dans le framework, dans la classe application définie dans le script `core/om_application.class.php` l'attribut `$html_head_favicon`. Actuellement le framework ne spécifie aucun favicon par défaut.

```
<?php
...
class application {

    /**
     * Gestion du favicon de l'application.
     *
     * @var mixed Configuration niveau framework.
     */
    var $html_head_favicon = null;
...
?>
```

Pour récupérer la valeur du paramètre sans se préoccuper d'où vient le paramètre l'accesseur `application::get_config__favicon()` est disponible. C'est toujours cette méthode qui doit être utilisée pour accéder au paramètre. Exemple d'utilisation :

```
<?php
...
$f->get_config__favicon();
...
?>
```

4.3.7.8 Le mode de gestion des permissions

Ce paramètre permet de définir si la gestion des profils se fait de manière hiérarchique ou non. Si on décide d'utiliser les profils hiérarchiques alors un utilisateur qui a le profil SUPER UTILISATEUR (hiérarchie 4) peut effectuer toutes les actions possibles pour un utilisateur qui a le profil UTILISATEUR (hiérarchie 3). Par contre si on décide d'utiliser les profils non hiérarchiques, l'utilisateur qui a le profil SUPER UTILISATEUR ne peut effectuer que les actions qui lui sont permises spécifiquement. Important : la modification de cette option doit être suivie de la modification complète du paramétrage des droits.

Trois niveaux de configuration sont disponibles pour cet élément : framework, application et instance. Voici l'ordre de préférence si les trois niveaux sont configurés : instance > application > framework.

Pour configurer au niveau de l'instance, il faut définir dans le script `dyn/config.inc.php` le paramètre **permission_by_hierarchical_profile** sur le tableau `$config`.

```
<?php
$config = array();
$config["permission_by_hierarchical_profile"] = true;
?>
```

Pour configurer au niveau de l'application, il faut définir dans la classe `utils` définie dans le script `obj/utils.class.php` l'attribut `$config__permission_by_hierarchical_profile`.

```
<?php
...
class utils extends application {

    /**
     * Gestion du mode de gestion des permissions.
     *
     * @var mixed Configuration niveau application.
     */
    var $config__permission_by_hierarchical_profile = false;
...
?>
```

Une configuration par défaut est définie dans le framework, dans la classe `application` définie dans le script `core/om_application.class.php` l'attribut "`config__permission_by_hierarchical_profile`".

```
<?php
...
class application {

    /**
     * Gestion du mode de gestion des permissions.
     *
     * @var mixed Configuration niveau framework.
     */
    var $config__permission_by_hierarchical_profile = true;
...
?>
```

Pour récupérer la valeur du paramètre sans se préoccuper d'où vient le paramètre l'accesseur `application::get_config__permission_by_hierarchical_profile()` est disponible. C'est toujours cette méthode qui doit être utilisée pour accéder au paramètre. Exemple d'utilisation :

```
<?php
...
$f->get_config__permission_by_hierarchical_profile();
...
?>
```

4.3.7.9 La valeur par défaut lorsqu'une permission n'existe pas

Ce paramètre permet de spécifier la valeur retour de la méthode vérifiant si l'utilisateur possède une permission lorsque cette permission n'existe pas. Ce paramètre est défini au niveau du framework à la valeur *false* ce qui signifie que si la permission n'existe pas alors la méthode va retourner que l'utilisateur n'a pas la permission. Ce paramètre peut éventuellement être surchargé au niveau de l'instance. Il suffit de définir dans le script `dyn/config.inc.php` le paramètre **`permission_if_right_does_not_exist`** sur le tableau `$config`. Important : il est conseillé de ne surcharger ce paramètre que sur une instance de développement et jamais en production.

```
<?php
$config = array();
$config["permission_if_right_does_not_exist"] = true;
?>
```

4.3.7.10 Les extensions de fichiers autorisées

Utilisé dans le module upload.php. Chaque extension est séparée avec un «;».

```
<?php
$config = array();
$config["upload_extension"] = ".gif;.jpg;.jpeg;.png;.txt;.pdf;.csv;";
?>
```

4.3.7.11 La taille maximale de fichiers autorisée

Utilisé dans le module upload.php. La taille maximale est en mo.

```
<?php
$config = array();
$config["upload_taille_max"] = str_replace('M', '', ini_get('upload_max_filesize')) *
↳1024;
?>
```

4.3.8 Le Parametrage des librairies

Le paramétrage de l'accès aux librairies se fait dans *dyn/include.inc.php*

Ce fichier permet de configurer les paths en fonction de la directive `include_path` du fichier `php.ini`. Vous pouvez aussi modifier ces chemins avec vos propres valeurs si vous voulez personnaliser votre installation :

PEAR

```
array_push($include, getcwd()."/../php/pear");
```

DB

```
array_push($include, getcwd()."/../php/db");
```

FPDF

```
array_push($include, getcwd()."/../php/fpdf");
```

OPENMAIRIE (dans CORE depuis la version 4.2.0)

```
define("PATH_OPENMAIRIE", getcwd()."/../core/openmairie/");
```

Par défaut, les librairies sont incluses dans `openmairie_exemple` :

- /lib : contient les librairies javascript
- /php : contient les librairies php

4.3.9 Le mode DEBUG

Dans le code, pour logger une information, il suffit de d'utiliser la ligne suivante :

```
$this->addLog("requete sig_interne maj parcelle inexistante :".$sql, EXTRA_VERBOSE_
↪MODE);
```

Les différents modes DEBUG présents dans l'application sont définis dans le fichier *core/om_debug.inc.php* :

- EXTRA_VERBOSE_MODE - mode « très bavard » : affiche tous les messages
- VERBOSE_MODE - mode « bavard » : affiche tous les messages d'erreur ainsi que toutes les requêtes exécutées
- DEBUG_MODE - mode « debug » : affiche tous les messages d'erreur
- PRODUCTION_MODE - mode « production » : il n'y a pas de message

Dans l'applicatif, on peut paramétrer le mode DEBUG grâce au fichier *dyn/debug.inc.php* suivant. Il suffit de commenter/décommenter l'instruction définie souhaitée.

```
<?php
/**
 * Ce fichier contient le parametrage pour le mode debug
 *
 * @package openmairie_exemple
 * @version SVN : $Id: debug.inc.php 2198 2013-03-28 17:08:33Z fmichon $
 */

/**
 *
 */

(define("PATH_OPENMAIRIE") ? "" : define("PATH_OPENMAIRIE", ""));
require_once PATH_OPENMAIRIE."om_debug.inc.php";

/**
 *
 */
//define('DEBUG', EXTRA_VERBOSE_MODE);
//define('DEBUG', VERBOSE_MODE);
//define('DEBUG', DEBUG_MODE);
define('DEBUG', PRODUCTION_MODE);

?>
```

Dans le dossier tmp/error_log.txt les messages de logs de niveau DEBUG_MODE sont écrits quelque soit le mode définit.

4.3.10 La version de votre application

Vous devez mettre le numéro de version et la date de votre application dans *dyn/version.inc*

Voir *le versionage des applications*.

4.3.11 Les informations generales

Les fichiers textes d'information générale sont à la racine de l'application :

README.txt :

ce fichier peut contenir entre autre, la liste des auteurs ayant participé au projet

HISTORY.txt : information sur chaque version :

les (+) et les (bugs) corrigés

app/SPECIFIC.txt :

Ici, vous décrivez la spécificité de l'application courante par rapport au framework

LICENCE.txt : licence libre de l'application

TODO.txt : feuille de route - roadmap

INSTALL.txt : installation de l'application

4.3.12 L'installation automatique

Un fichier data/sql/install.sql permet d'installer rapidement et data/sql/make_init.sh permet de constituer rapidement des scripts sql d'installation.

4.3.13 Les paramètres des combos

Les paramètres des combos sont paramétrés dans les fichiers suivants (type de contrôle de formulaire comboD et comboG (pour formulaire) ou comboD2 et comboG2 (pour sous formulaire))

```
- comboaffichage.inc.php :  
    paramètre de l'affichage dans la fenêtre combo.php  
- comboparametre.inc.php  
    affecte des valeurs spécifiques au formulaire parent si il y a plusieurs  
    enregistrement en lien (choix en affichage)  
- comboretour.inc.php  
    même chose que comboparametre.inc si il n'y a qu'un enregistrement en lien  
    (pas d'affichage de la fenêtre)
```

Voir *chapitre framework/formulaire, sous programme générique combo.php*

4.3.14 Les paramètres éditions

Les variables dans les éditions sont paramétrées dans

```
- varpdf.inc                pour les pdf  
- varetatpdf.inc           pour les états et les sous états  
- varlettretypetpdf.inc    pour les lettres type
```

Voir *chapitre framework/édition*

4.3.15 Les paramètres om_sig

var_sig.php

les paramètres sont les suivants

```
$contenu_etendue[0]= array('4.5868,43.6518,4.6738,43.7018'  
                           );  
$contenu_etendue[1]= array('vitrolles'  
                           );  
$contenu_epsg[0] = array("", "EPSG:2154", "EPSG:27563");
```

(suite sur la page suivante)

(suite de la page précédente)

```
$contenu_epsg[1] = array("choisir la projection", 'lambert93', 'lambertSud');
$type_geometrie[0] = array("", "point", "line", "polygon");
$type_geometrie[1] = array("choisir le type de géométrie", 'point', 'ligne', 'polygone');
```

ces paramètres sont utilisés pour la saisie de carte : voir chapitre sig

Les etendues ne sont plus gérées dans ce fichier avec la version 4.5

Les post traitements de form_sig permettent de faire des traitement apres saisie de géométries avec om_sig

form_sig_update.inc.php

form_sig_delete.inc.php

Ces post traitements sont deprecated en version 4.5.

4.4 La gestion des accès

Sommaire

- *La gestion des accès*
- *Introduction*
- *Fonctionnalités*
- *Les tables*
- *Les règles*
- *La multi-collectivité*
- *Les login et logout*
- *Les utilitaires*

4.4.1 Introduction

Le framework fournit un gestionnaire d'accès configurable dans les menus :

- administration -> profil
- administration -> droit
- administration -> utilisateur

Les accès sont conservés dans les tables du même nom.

4.4.2 Fonctionnalités

- Synchronisation des utilisateurs provenant d'un annuaire
- Modification du mot de passe par l'utilisateur
- Mot de passe oublié et réinitialisation du mot de passe

4.4.3 Les tables

La gestion des accès est gérée avec 3 tables :

om_profil : gestion par défaut de 5 profils :

- administrateur (5)

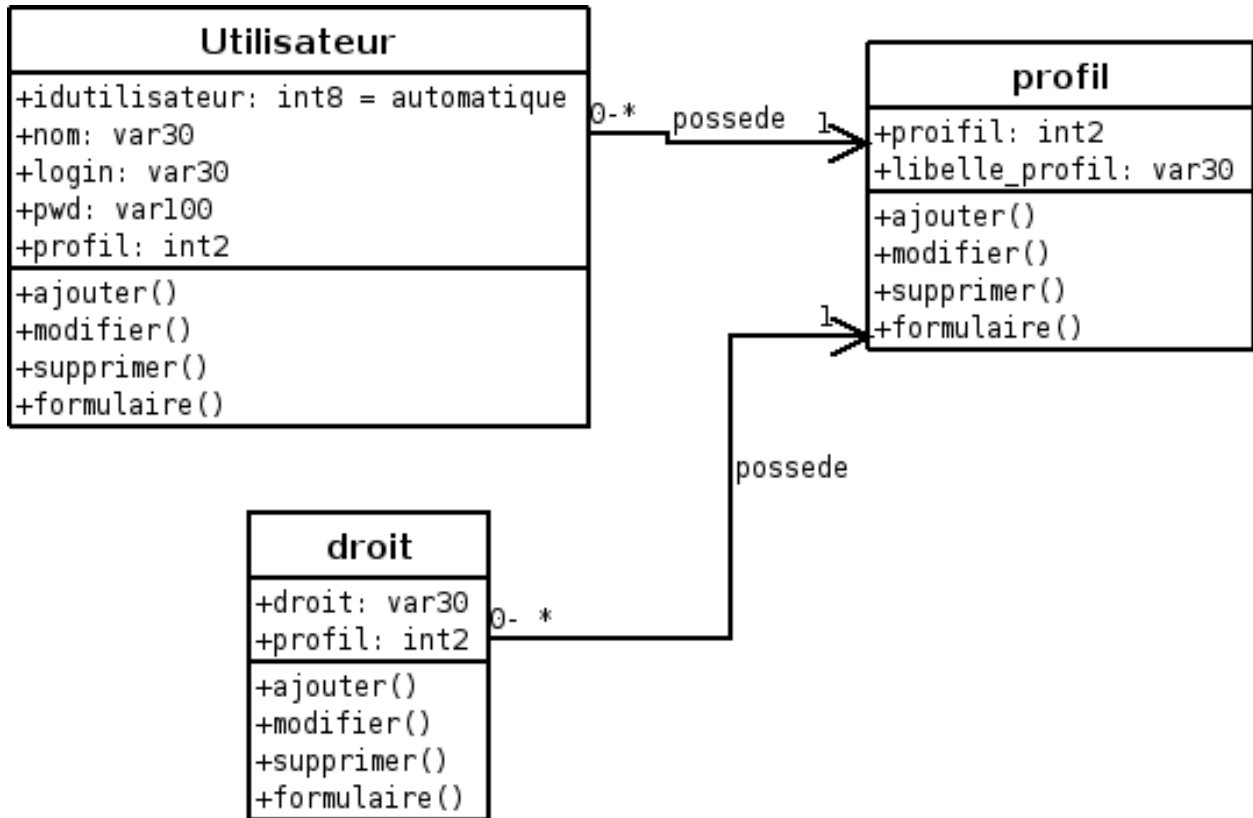
- super utilisateur (4)
- utilisateur (3)
- utilisateur limité (2)
- consultation (1)

Les profils sont hiérarchiques, le profil 5 étant le plus élevé, il a accès à toutes les actions des profils inférieurs.

om_droit : à chaque profil est affecté un ou plusieurs droits.

om_utilisateur : cette table permet de donner un login, un mot de passe et un profil à chaque utilisateur.

Diagramme de classe



4.4.4 Les règles

Le droit sur un objet porte le nom de l'objet, pour chaque objet il existe deux types de droits :

- généraux : il n'est composé que du nom de l'objet et permet d'accéder à toutes les actions sur celui-ci.
- spécifique : il se compose du nom de l'objet puis d'un suffixe.

Détails des suffixes de droits :

- `_tab` : permet d'accéder au tableau
- `_ajouter` : permet d'ajouter un objet
- `_modifier` : permet de modifier l'objet
- `_supprimer` : permet de supprimer l'objet
- `_consulter` : permet de consulter l'objet

4.4.5 La multi-collectivité

Les collectivités peuvent être de niveau 1 ou de niveau 2. Les utilisateurs de chaque collectivité héritent de ce niveau. Les utilisateurs de niveau 1 n'ont accès qu'à leur collectivité tandis que les utilisateurs de niveau 2 ont accès à toutes les collectivités disponibles. Lors de la conception de la base de données un champ `om_collectivite` peut être ajouté à chaque table ayant besoin d'un filtrage par collectivité. Les utilisateurs de niveau 1 ne verront aucune notion de collectivité et n'auront accès qu'aux éléments liés à leur propre collectivité.

4.4.6 Les login et logout

Le login se fait par le script `scr/login.php`

`login.php` valorise les variables sessions permettant la gestion des accès et sécurités

```
<?php
$_SESSION['profil'] = $profil;
$_SESSION['nom'] = $nom;
$_SESSION['login'] = $login;
?>
```

La déconnexion se fait avec le script `scr/logout.php`

Le changement de mot de passe se fait avec le script `scr/password.php`

L'accès au changement de mot de passe se fait par défaut dans le menu haut (voir `framework/paramétrage`)

4.4.7 Les utilitaires

La gestion des droits d'accès se fait dans les méthodes des utilitaires :

- `php/openmairie/om_application.class.php` (composant openMairie)
- `obj/utils.class.php`

(voir `framework/utilitaire`)

4.5 Le tableau de bord

Sommaire

- *Le tableau de bord*
 - *Principe*
 - *paramétrage*
 - *les widgets*
 - *le tableau de bord paramétrable*
 - *widget*
 - *la création de widget*
 - *Le widget de type "Web"*
 - *interne*
 - *externe*
 - *Le widget de type "Script"*
 - *Modèle de données*
 - *Les tableaux de bord*
 - *accès au tableau de bord*

— *Modèle de données*
— *Composants*

4.5.1 Principe

Il est proposé dans ce chapitre de décrire le tableau de bord paramétrable pour les utilisateurs

4.5.1.1 paramétrage

l'accès au tableau de bord paramétrable dyn/dashboard.inc.php

(voir framework/paramétrage)

Par défaut, le tableau de bord paramétrable est activé, il peut être déconnecté en enlevant le commentaire // die().

4.5.1.2 les widgets

Les widgets sont des liens et/ou de petits scripts paramétrables qui peuvent être rajoutés dans le tableau de bord. Ces scripts sont conservés dans la table om_widget.

Chaque utilisateur paramètre son tableau de bord.

4.5.1.3 le tableau de bord paramétrable

L'administrateur choisit les widgets présents sur le tableau de bord de chaque profil parmi ceux proposés dans l'application. Il peut placer les widgets où il le souhaite.

4.5.2 widget

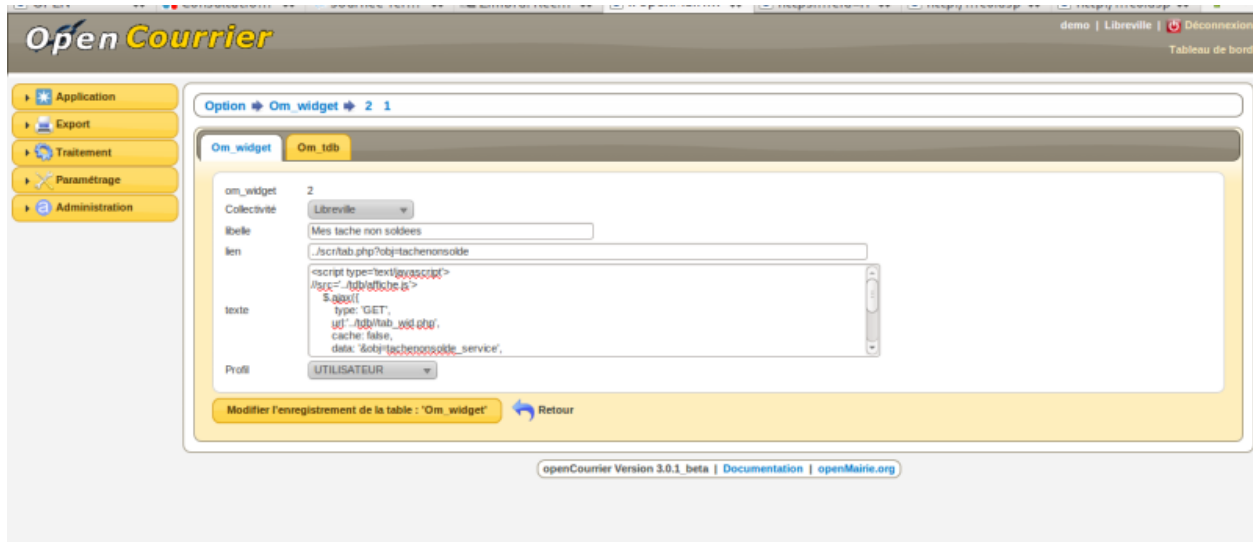
Le widget (WIDGET DASHBOARD) est un bloc d'informations contextualisées accessible depuis le tableau de bord de l'utilisateur. Il peut être de type "Web" ou de type "Script".

4.5.2.1 la création de widget

La saisie des widget se fait dans administration -> om_widget.

La grille de saisie est la suivante

```
libellé du widget qui apparaîtra à l'ajout du widget dans le tableau de bord
lien qui sera implémenté (# : pas de lien)
texte : texte du widget (iframe, javascript, ajax ...)
profil : profil autorisé pour le tableau de bord
```



Le tableau de bord, peut gérer toutes sortes d'informations internes ou externes à l'application

```
les taches non soldees pour openCourrier
les appels à la maintenance
l'horoscope, la météo, une vidéo, des photos ...
```

4.5.2.2 Le widget de type "Web"

4.5.2.2.1 interne

les liens sur les cartes (à mettre dans le champ lien) :

```
la carte de raphele avec tab_sig_point.php
../scr/tab_sig_point_db.php?obj=raphele_1&zoom=6
celle de mas thibert :
../scr/tab_sig_point.php?obj=odp_6&zoom=7
```

les accès personnalisés « ajax » au travers de son code utilisateur (dans openCourrier)

```
<script type='text/javascript'>
$.ajax({
  type: 'GET',
  url: '../app/tab_wid.php',
  cache: false,
  data: '&obj=tachenonsolde_service',
  success: function(html) {
    $('#aff3').append(html);
  }
});
</script>
<div id='aff3'></div>
```

Ce code lance dans le widget `../app/tab_wid.php?obj=tachenonsolde_service`

`tachenonsolde_service` est initialisé dans `sql/mysql/tachenonsolde_service.inc`

Il ne s'affichera que la première page (paramétrer \$serie pour le nombre d'enregistrement affichés)

Attention si vous affichez plusieurs widgets « openmairie », mettre un id différent pour chaque div (ici aff3)

4.5.2.2.2 externe

Les autres applications openMairie peuvent aussi être accessibles par widget de la même manière que le paragraphe ci dessus.

D'autres widgets externes sont accessibles en mettant dans le champ texte les scripts suivants :

Acces à une video externe avec un « iframe »

```
<iframe width='200' height='150'
  src='http://www.youtube.com/embed/gS5B4LlqkFI'
  frameborder='0' allowfullscreen>
</iframe>
```

La meteo grace à un javascript du site tameteo.com

```
<div id='cont_f5089b722555454d1872b91f52beafd4'>
  <h2 id='h_f5089b722555454d1872b91f52beafd4'>
  <a href='http://www.tameteo.com/' title='Météo'>Météo</a></h2>
  <a id='a_f5089b722555454d1872b91f52beafd4'
    href='http://www.tameteo.com/
      meteo_Arles-Europe-France-Bouches+du+Rhone--1-25772.html'
    target='_blank' title='Météo Arles'
    style='color:#666666;font-family:1;font-size:14px;'></a>
  <script type='text/javascript'
    src='http://www.tameteo.com/wid_loader/f5089b722555454d1872b91f52beafd4'>
  </script>
</div>
```

Horoscope au travers d un iframe qui pointe sr astroo.com

```
<!--DEBUT CODE ASTROO-->
<!--debut code perso-->
<iframe width='232' height='302' marginheight='0' marginwidth='0' frameborder='0'
  align='center' src='http://www.astroo.com/horoscope.htm'
  name='astroo' allowtransparency='true'>
<!--fin code perso-->
<a href='http://www.astroo.com/horoscope.php' target='_top'
  title='Cliquez-ici pour afficher l'horoscope quotidien'>
  <font face='Verdana' size='2'><b>afficher l'horoscope du jour</b>
  </font></a>
</iframe>
<noscript>
<a href='http://www.astroo.com/horoscope.php' target='_blank'>horoscope</a>
</noscript>
<!--FIN CODE ASTROO-->
```

Acces à un fil rss avec un module ajax google

```
<script src='http://www.gmodules.com/ig/ifr?url=
  http://www.ajaxgaier.com/iGoogle/rss-reader%2B.xml
  &up_title=Actualit%C3%A9s%20atReal
  &up_feed=http%3A%2F%2Fwww.atreal.fr%2Fatreal%2Fcommunaute%2Factualites-atreal%2FRSS
  &up_contentnr=9&up_fontsize=9&up_lineheight=70
  &up_titlelink=&up_bullet=1
  &up_reload_feed=0&up_reload_fcqcy=0
  &up_hl_background=FFFFFF&synd=open&w=200&h=100
  &title=
```

(suite sur la page suivante)

(suite de la page précédente)

```
&border=%23ffffff%7C3px%2C1px+solid+%23999999&output=js'>
</script>
```

Affichage de photos avec flick “r (appel javascript) :

```
<table><tr>
<div class='flickr_r'>
<script type='text/javascript'
    src='http://www.flickr.com/badge_code_v2.gne?count=3
        &display=latest&size=s
        &layout=h&source=user
        &user=27995901%40N03'></script>
</div>
</tr></table>
```

4.5.2.3 Le widget de type “Script”

app/widget_example.php

```
<?php
/**
 * WIDGET DASHBOARD - widget_example.
 *
 * L'objet de ce script est de fournir un exemple de widget de type 'Script'.
 *
 * @package openmairie_framework
 * @version SVN : $Id$
 */

// On instancie la classe utils uniquement si la variable $f n'est pas déjà définie
// pour protéger l'accès direct au script depuis l'URL. La permission "forbidden"
// a pour vocation de n'être donnée à aucun utilisateur.
require_once "../obj/utils.class.php";
if (!isset($f)) {
    $f = new utils(null, "forbidden");
}

//
$footer = "";

//
$footer_title = "";

//
$widget_is_empty = true;

?>
```

4.5.2.4 Modèle de données

```
CREATE TABLE om_widget
(
    om_widget integer NOT NULL, -- Identifiant unique
```

(suite sur la page suivante)

(suite de la page précédente)

```
libelle character varying(100) NOT NULL, -- Libellé du widget
lien character varying(80) NOT NULL DEFAULT ' '::character varying, -- Lien qui_
↳ pointe vers le widget (peut être vers une URL ou un fichier)
texte text NOT NULL DEFAULT ' '::text, -- Texte affiché dans le widget
↳ type character varying(40) NOT NULL, -- Type du widget ('web' si pointe vers une_
↳ URL ou 'file' si pointe vers un fichier)
CONSTRAINT om_widget_pkey PRIMARY KEY (om_widget)
);
```

```
— obj/om_widget.class.php
— sql/pgsql/om_widget.form.inc.php
— sql/pgsql/om_widget.inc.php
— core/obj/om_widget.class.php
— core/sql/pgsql/om_widget.form.inc.php
— core/sql/pgsql/om_widget.inc.php
— gen/obj/om_widget.class.php
— gen/sql/pgsql/om_widget.form.inc.php
— gen/sql/pgsql/om_widget.inc.php
```

4.5.3 Les tableaux de bord

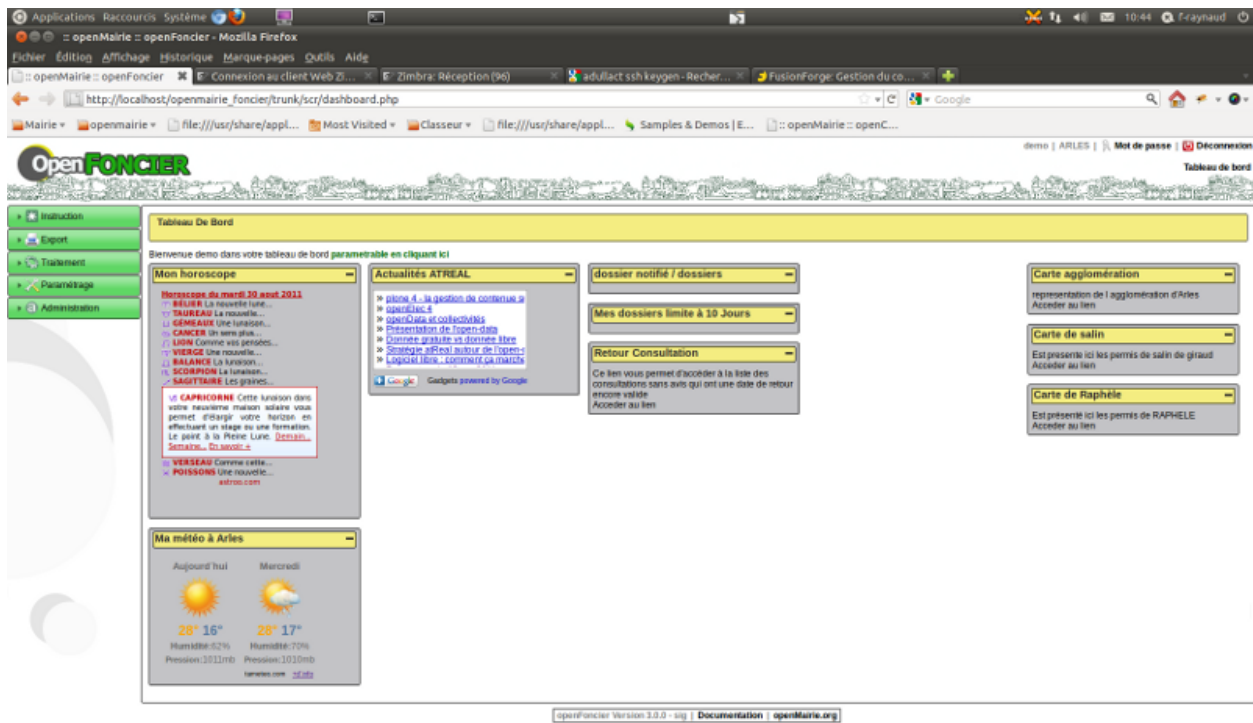
4.5.3.1 accès au tableau de bord

Le paramétrage se fait en cliquant sur le lien « paramétrer son tableau de bord »

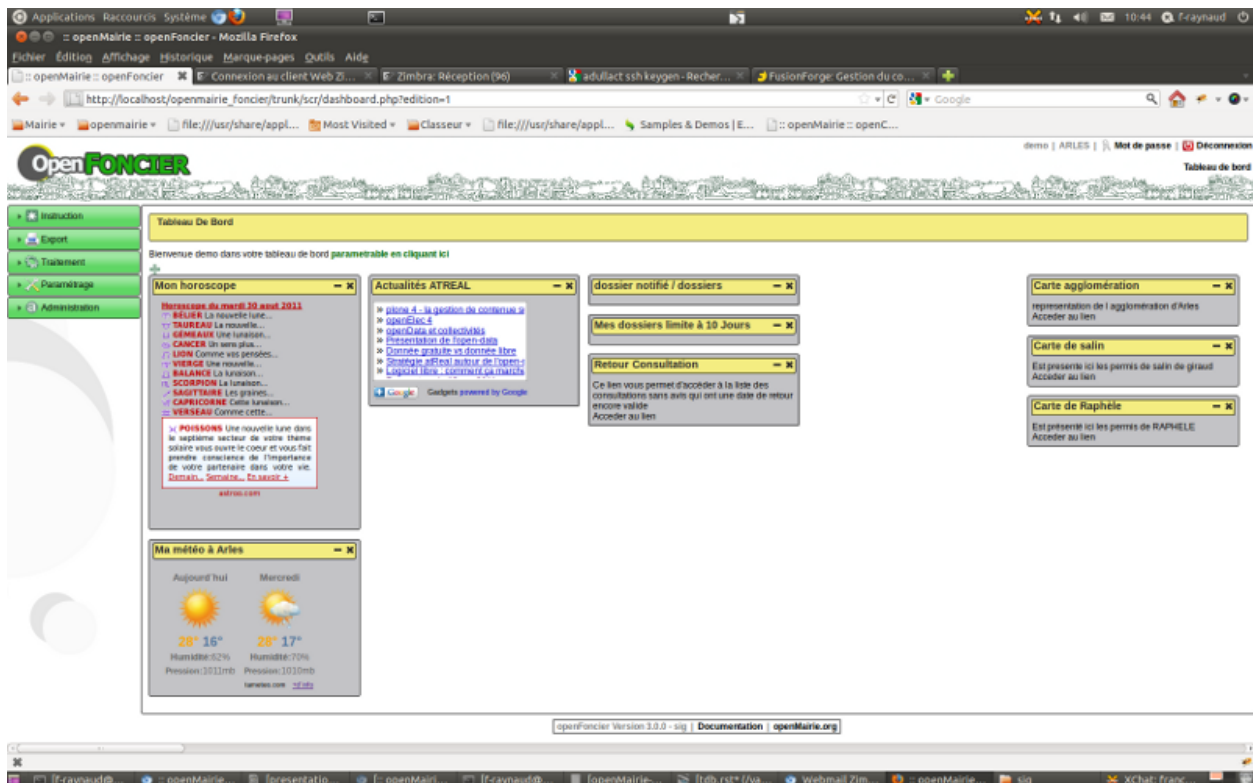
Il apparaît alors

```
un "plus" pour ajouter un widget pour une colone
une croix pour supprimer un widget
```

Le déplacement du widget de haut en bas ou de gauche à droite se fait par copier/glisser avec la souris.



En cliquant sur « + », il est possible de rajouter des widgets dans son tableau de bord



Dans la version om 4.5, la composition du tableau de bord se fait avec l'option « composer le tableau de bord » du menu administration. Le paramétrage se fait par profil.

4.5.3.2 Modèle de données

```
CREATE TABLE om_dashboard
(
  om_dashboard integer NOT NULL, -- Identifiant unique
  om_profil integer NOT NULL, -- Profil auquel on affecte le tableau de ville
  bloc character varying(10) NOT NULL, -- Bloc de positionnement du widget
  "position" integer, -- Position du widget dans le bloc
  om_widget integer NOT NULL, -- Identifiant du widget
  CONSTRAINT om_dashboard_pkey PRIMARY KEY (om_dashboard),
  CONSTRAINT om_dashboard_om_profil_fkey FOREIGN KEY (om_profil)
    REFERENCES openexemple.om_profil (om_profil),
  CONSTRAINT om_dashboard_om_widget_fkey FOREIGN KEY (om_widget)
    REFERENCES openexemple.om_widget (om_widget)
);
```

```
— obj/om_dashboard.class.php
— sql/pgsql/om_dashboard.form.inc.php
— sql/pgsql/om_dashboard.inc.php
— core/obj/om_dashboard.class.php
— core/sql/pgsql/om_dashboard.form.inc.php
— core/sql/pgsql/om_dashboard.inc.php
— gen/obj/om_dashboard.class.php
— gen/sql/pgsql/om_dashboard.form.inc.php
— gen/sql/pgsql/om_dashboard.inc.php
```

4.5.4 Composants

Les scripts du framework qui s'occupent de la gestion du tableau de bord sont :

```
— scr/dashboard.php
— scr/dashboard_composer.php
— spg/widgetctl.php
```

4.6 Les listings

Sommaire

```
— Les listings
— Introduction
— Les scripts scr/tab.php et scr/soustab.php
— Configuration
  — $ent
  — $serie
  — $table
  — $champAffiche
  — $champRecherche
  — $tri
  — $selection
  — $edition
  — $tab_title
```

- *\$tab_description*
- *\$tab_actions*
- *Actions des tableaux*
 - *Les actions par défaut*
 - *Créer de nouvelles actions*
 - *Définition de l'action*
 - *Définition du mode d'affichage en sous-tableau*
 - *Définition de l'ordre d'affichage*
 - *Définition des droits d'affichage*
- *Les fonctionnalités*
 - *Le tri*
 - *Premier clic pour le tri croissant*
 - *Second clic pour le tri décroissant*
 - *Troisième clic pour aucun tri particulier*
 - *L'export CSV*
 - *La recherche avancée*
 - *Les différents types de recherche*
 - *Recherche simple*
 - *Recherche avancée*
 - *Recherche avancée mono-critère*
 - *Recherche avancée multi-critères*
 - *Configuration de la recherche avancée*
 - *Activation*
 - *Autres paramètres*
 - *Configuration des critères de recherche*
 - *Configuration simple*
 - *Configuration avancée*
 - *Créer un intervalle de date*
 - *Créer un champ de recherche avec menu déroulant personnalisé*
 - *Tester si une donnée est présente ou non dans un groupe de données*
- *Le composant openMairie*

4.6.1 Introduction

Il est décrit dans ce paragraphe, l'utilisation et la configuration des tableaux d'enregistrements issus de la base de données.

La gestion des tableaux se base sur le fichier `core/om_table.class.php` (classe table)

Pour chaque affichage de tableau, le script `sql/DBTYPE/[objet].inc.php` correspondant est appelé. Il permet de stocker le détail des requêtes nécessaires à l’affichage du contenu.

Les tableaux sont construits lors de l’appel aux scripts `scr/tab.php` et `scr/soustab.php`.

4.6.2 Les scripts `scr/tab.php` et `scr/soustab.php`

L’appel à ces scripts permet d’afficher un tableau d’enregistrements de l’objet passé en paramètre.

Liste des paramètres passés à l’url :

- `obj` : nom de l’objet pour lequel on souhaite afficher le tableau
- `premier` : numéro de la première ligne affichée dans le tableau
- `recherche` : chaîne de caractères recherchée depuis le modules de recherche
- `selectioncol` : numéro de la colonne sélectionnée dans le module de recherche
- `tricol` : numéro de colonne et orientation (+/-) du tri du tableau
- `valide` : (true/false) affiche ou non les enregistrements non valide

L’appel à `scr/soustab.php` est fait en javascript depuis un formulaire afin d’afficher les informations liées à l’enregistrement en cours d’édition.

4.6.3 Configuration

Un listing se configure via le script `sql/pgsql/<OBJ>.inc.php`.

4.6.3.1 \$ent

Titre (fil d'ariane) de la page.

```
<?php
$ent = _("administration")." -> "._("om_parametre");
?>
```

4.6.3.2 \$serie

Nombre d'enregistrements par page.

```
<?php
$serie = 15;
?>
```

4.6.3.3 \$table

Table de référence (il peut y avoir une ou plusieurs jointure). Clause FROM de la requête du listing.

```
<?php
$table = DB_PREFIXE."om_parametre";
?>
```

4.6.3.4 \$champAffiche

Liste des champs du tableau

```
<?php
$champAffiche = array(
    'om_parametre',
    'libelle',
    'valeur',
    'om_collectivite',
);
?>
```

4.6.3.5 \$champRecherche

Champs pour la recherche.

```
<?php
$champRecherche = array(
    'libelle',
    'valeur',
);
?>
```

4.6.3.6 \$tri

Critère de tri par défaut.

```
<?php
$tri = " ORDER BY libelle ";
?>
```

4.6.3.7 \$selection

Gestion de la clause WHERE de la requête du listing.

```
<?php
$selection = "";
?>
```

4.6.3.8 \$edition

Édition PDF.

```
<?php
$edition = "om_parametre";
?>
```

4.6.3.9 \$tab_title

Titre de l'onglet du listing. Si cette valeur n'est pas définie alors c'est <OBJ> qui est utilisé ou plutôt la traduction de <OBJ>.

```
<?php
$tab_title = _("paramètre");
?>
```

4.6.3.10 \$tab_description

Description de la page. Si cette valeur n'est pas définie alors aucune description n'apparaît.

```
<?php
$tab_description = _("Ce listing présente tous les paramètres spécifiques à l
↪'utilisateur connecté.");
?>
```

4.6.3.11 \$tab_actions

Voir le paragraphe dédié : *Actions des tableaux*.

4.6.4 Actions des tableaux

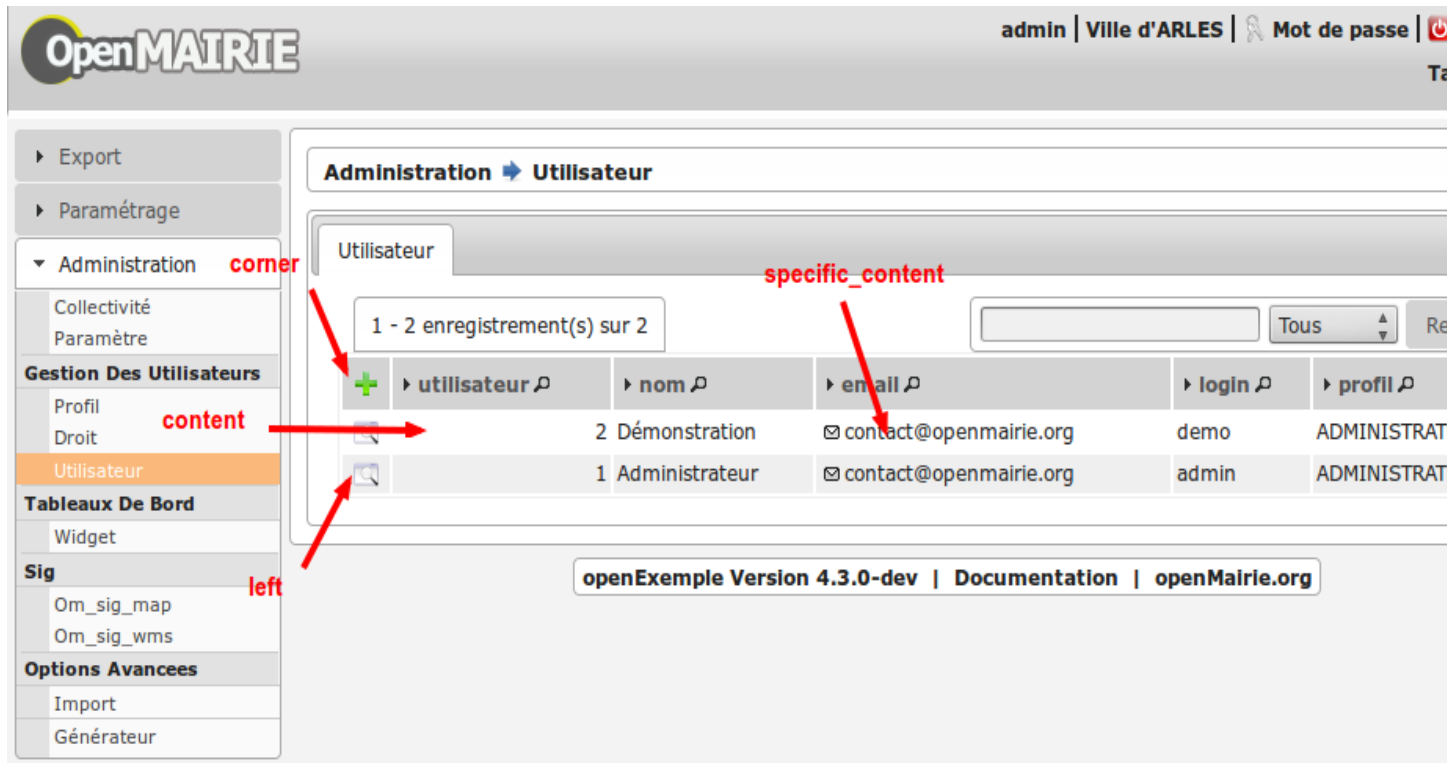
La surcharge des actions de tableaux se fait via les scripts `sql/sghd/objet.inc.php`.

L'ajout d'actions se présente de cette façon :

```
<?php
// Actions en coin ('corner') : ajouter
$tab_actions['corner']['ajouter'] = array(
    'lien' => 'form.php?obj='.$obj.'&action=0',
    'id' => '&advs_id='.$advs_id.'&tricol='.$tricol.'&valide=' .
    ↪ $valide.'&retour=tab',
    'lib' => '<span class="om-icon om-icon-16 om-icon-fix add-16" title="'._(
    ↪ 'Ajouter').'">'._('Ajouter').'</span>',
    'rights' => array('list' => array($obj, $obj.'_ajouter'), 'operator' =>
    ↪ 'OR'),
    'ordre' => 10,
);
// Actions à gauche ('left') : consulter
$tab_actions['left']['consulter'] = array(
    'lien' => 'form.php?obj='.$obj.'&action=3'.'&idx=',
    'id' => '&premier='.$premier.'&advs_id='.$advs_id.'&
    ↪ recherche='.$recherche.'&tricol='.$tricol.'&selectioncol=' .
    ↪ $selectioncol.'&valide='.$valide.'&retour=tab',
    'lib' => '<span class="om-icon om-icon-16 om-icon-fix consult-16" title="
    ↪ '._('Consulter').'">'._('Consulter').'</span>',
    'rights' => array('list' => array($obj, $obj.'_consulter'), 'operator' =>
    ↪ 'OR'),
    'ordre' => 10,
);
// Action sur la cinquième colonne de contenu
$tab_actions['specific_content'][4] = array(
    'lien' => 'form.php?obj='.$obj.'&action=2'.'&idx=',
    'id' => '&premier='.$premier.'&advs_id='.$advs_id.'&
    ↪ recherche='.$recherche.'&tricol='.$tricol.'&selectioncol=' .
    ↪ $selectioncol.'&valide='.$valide.'&retour=tab',
    'lib' => '<span class="om-icon om-icon-16 om-icon-fix delete-16" title="
    ↪ '._('Consulter').'">'._('Consulter').'</span>',
    'rights' => array('list' => array($obj, $obj.'_consulter'), 'operator' =>
    ↪ 'OR'),
    'ordre' => 10,
);
?>
```

Plusieurs emplacements d'actions existent :

- corner : actions dans la première cellule du tableau
- left : action situées dans la première colonne, disponibles pour chaque élément du tableau
- content : action sur le contenu du tableau
- specific_content : action sur une colonne de contenu du tableau



4.6.4.1 Les actions par défaut

Par défaut seules les actions ajouter et consulter sont disponibles depuis les tableaux.

4.6.4.2 Créer de nouvelles actions

La création d'actions pour un tableau particulier se fait depuis le répertoire `sql/sqbd/`.

Les actions doivent se définir dans les fichier `objet.inc.php` de la manière suivante :

```
<?php
$tab_actions['left']['modifier'] = array(
    'lien' => 'form.php?obj='.$obj.'&action=1'.'&idx=',
    'id' => '&premier='.$premier.'&adv_id='.$adv_id.'&recherche='.
    $recherche.'&tricol='.$tricol.'&selectioncol='.$selectioncol.'&valide=
    $valide.'&retour=tab',
    'lib' => '<span class="om-icon om-icon-16 om-icon-fix edit-16" title="'.__(
    'Modifier').">'.__('Modifier').</span>',
    'rights' => array('list' => array($obj, $obj.'_modifier'), 'operator' => 'OR'),
    'ordre' => 20,
);
?>
```

4.6.4.2.1 Définition de l'action

La première clé de `$tab_actions` permet choisir la position d'affichage :

- `corner` pour les actions en coin ;
- `left` pour les actions de gauche.

Note : Depuis la version 4.3.0 d'openMairie, il est désormais possible d'afficher plusieurs actions dans le coin du tableau (au niveau de l'action `ajouter`).

La seconde clé de `$tab_actions` permet de définir la nouvelle action. Cette clé doit être différente de : `ajouter`, `consulter`, `modifier` et `supprimer`.

Les clés `lien`, `id` et `lib` s'utilisent de la même manière qu'avant.

4.6.4.2.2 Définition du mode d'affichage en sous-tableau

La clé `ajax` permet d'indiquer si l'action doit être affichée en ajax ou non dans les sous-tableaux :

- `true`, l'action utilisera la fonction `ajaxIt()` ;
- `false`, l'action n'utilisera pas la fonction `ajaxIt()`.

4.6.4.2.3 Définition de l'ordre d'affichage

La clé `ordre` permet de déterminer l'ordre d'affichage par rapport aux autres actions.

Chaque action dispose d'une valeur numérique permettant de définir sa place au sein d'une position. L'action numéro 1 s'affichera en premier, l'action numéro 10 s'affichera après les actions de numéro inférieur, etc.

Ordre des actions par défaut d'openMairie :

- `ajouter` à pour ordre 10 dans la position `corner` ;
- `consulter` à pour ordre 10 dans la position `left`.

Si la position `corner` est sélectionnée :

- 9, l'action s'affichera avant l'action `ajouter` ;
- 11, l'action s'affichera après l'action `ajouter`.

Si la position `left` est sélectionnée :

- 9, l'action s'affichera avant l'action `consulter` ;
- 11, l'action s'affichera après l'action `consulter`.

4.6.4.2.4 Définition des droits d'affichage

La clé `rights` permet de définir le ou les droits nécessaires à l'utilisateur pour visualiser cette action. Cette clé est optionnelle. Si `rights` n'existe pas, tous les utilisateurs pourront visualiser cette action s'ils peuvent visualiser le tableau correspondant.

La clé `list` permet de définir le tableau des droits nécessaire.

La clé `operator` permet de définir l'opérateur utilisé pour vérifier les droits de la liste `list` :

- `OR`, l'utilisateur doit avoir au moins un droit ;
- `AND`, l'utilisateur doit avoir tous les droits.

4.6.5 Les fonctionnalités

- la recherche simple
- la recherche avancée
- la pagination
- le tri
- les éléments archivés
- l'export PDF

- l'export CSV
- les actions

4.6.5.1 Le tri

Par défaut le listing est trié en fonction du critère ORDER BY de la requête paramétrable via la variable \$tri dans le script `sql/<SGBD>/<OBJ>.inc.php` :

```
<?php
// Critère de tri par défaut
$tri = "";
// Exemple de tri su la colonne libellé de la table om_droit
$tri = " ORDER BY om_droit.libellé ";
?>
```

Une fois le listing chargé avec le tri par défaut, l'utilisateur peut choisir de trier sur une colonne en particulier en cliquant sur le titre de la colonne.

4.6.5.1.1 Premier clic pour le tri croissant

Marqueur représentant le tri croissant :

```
<span class="ui-icon ui-icon-triangle-1-s"><!-- --></span>
```



4.6.5.1.2 Second clic pour le tri décroissant

Marqueur représentant le tri décroissant :

```
<span class="ui-icon ui-icon-triangle-1-n"><!-- --></span>
```



4.6.5.1.3 Troisième clic pour aucun tri particulier

Marqueur représentant aucun tri particulier :

```
<span class="ui-icon ui-icon-triangle-1-e"><!-- --></span>
```



4.6.5.2 L'export CSV

L'export CSV peut être activé seulement si la recherche avancée est configurée sur le listing. Il suffit donc de rajouter dans le tableau de paramétrage de cette dernière la clé "export" avec la valeur array("csv",) comme le montre l'exemple suivant :

```
$options [] = array (
    'type' => 'search',
    'display' => true,
    'advanced' => $champs,
    'export' => array('csv',),
    'default_form' => 'advanced',
    'absolute_object' => 'facture'
);
```

Note : Par défaut l'export CSV reprend la requête SQL d'affichage définie précédemment. Le script de paramétrage `../sql/pgsql/<OBJ>.export_csv.inc.php` permet de surcharger ses paramètres.

Exemple d'utilisation : écraser `$champAffiche` pour redéfinir les colonnes du CSV exporté.

4.6.5.3 La recherche avancée

4.6.5.3.1 Les différents types de recherche

4.6.5.3.1.1 Recherche simple

Cette recherche est celle disponible par défaut sur les tableaux d'openMairie.

Elle permet de :

- rechercher une valeur dans une colonne parmi toutes celles affichées ;
- rechercher une valeur dans toutes les colonnes affichées ;
- rechercher des valeurs approximatives.

Note : Il est possible de modifier la liste des colonnes dans laquelle est effectuée la recherche. Cette liste ne correspond pas forcément aux colonnes affichées. Elle correspond seulement par défaut, c'est à dire lorsqu'aucune surcharge ne modifie les fichiers générés dans `gen/sql/`.

4.6.5.3.1.2 Recherche avancée

Cette recherche est une fonctionnalité qui peut être activée et configurée manuellement pour un ou plusieurs tableaux donnés.

Elle permet de :

- afficher un formulaire de recherche mono-critère permettant d'effectuer des recherches strictes ou approximatives ;
- afficher un formulaire de recherche multi-critères permettant d'effectuer des recherches strictes ou approximatives ;
- rechercher des valeurs dans des tables et des colonnes qui ne sont pas affichées.

Le numéro d'action (*\$maj*) consacré à la recherche avancée est le 999.

4.6.5.3.1.3 Recherche avancée mono-critère

Le formulaire de recherche mono-critère est un formulaire ne s'affichant que si la recherche avancée est activée. Il permet aux utilisateurs de basculer sur un formulaire similaire à celui de recherche simple lorsque la recherche avancée est activée.

Ce formulaire se comporte de la même manière que celui de recherche simple, avec quelques différences :

- il permet de rechercher des valeurs strictes ou approximatives (par défaut approximatives) ;
- il recherche dans toutes les colonnes proposées par la recherche simple ;
- il conserve les valeurs recherchées après la réalisation d'une action (ajout, modification, etc...) ;
- il dispose d'un bouton `Vider le formulaire` permettant de vider les champs ;
- il dispose d'un bouton `+` permettant de basculer sur le formulaire multi-critères.

4.6.5.3.1.4 Recherche avancée multi-critères

Le formulaire de recherche multi-critères est un formulaire ne s'affichant que si la recherche avancée est activée. Il permet aux utilisateurs de bénéficier de plusieurs champs, et ainsi effectuer des recherches plus précises qu'avec le

formulaire de recherche simple.

Description du formulaire :

- il peut afficher plusieurs champs, de type texte, nombre, date ou liste à choix ;
- il permet, pour chaque tableau, de configurer la liste des champs affichés ;
- il permet, pour chaque champ, de rechercher des valeurs strictes ou approximatives (par défaut approximatives) ;
- il permet, pour chaque champ, de rechercher des valeurs dans des tables et des colonnes qui ne sont pas affichées ;
- il conserve les valeurs recherchées après la réalisation d'une action (ajout, modification, etc...) ;
- il dispose d'un bouton `Vider le formulaire` permettant de vider les champs ;
- il dispose d'un bouton `+` permettant de basculer sur le formulaire mono-critère.

4.6.5.3.2 Configuration de la recherche avancée

4.6.5.3.2.1 Activation

Exemple avec le modèle `om_utilisateur`.

Pour activer la recherche avancée, rendez-vous dans le fichier `sql/sqbd/om_utilisateur.inc.php` et ajoutez la configuration suivante au tableau d'options :

```
<?php

$options[] = array('type' => 'search',
                  'display' => true,
                  'advanced' => $champs,
                  'default_form' => 'advanced',
                  'absolute_object' => 'om_utilisateur');

?>
```

Note : A partir de la version 4.3.0 d'openMairie, le tableau `$options` est disponible dans les fichiers `sql/` de l'application. Il n'est plus nécessaire de le déclarer manuellement.

La clé `type` est obligatoire. Elle permet de définir le type de l'option. Pour une recherche il faut saisir `search`.

La clé `display` est obligatoire. Elle permet d'afficher ou non la recherche, tout en conservant sa configuration.

- `true` permet d'afficher la recherche ;
- `false` permet de masquer la recherche.

La clé `advanced` est obligatoire (pour la recherche avancée). Elle permet de préciser que le formulaire de recherche est un formulaire de recherche avancée et non simple. Cette clé doit contenir le tableau des champs configurés pour la recherche (voir plus bas pour la configuration des champs).

La clé `default_form` est optionnelle. Elle permet de choisir quel formulaire de recherche est ouvert par défaut. La valeur `advanced` permet d'afficher le formulaire multi-critères. Les autres valeurs, ou si `default_form` n'est pas configuré, affichent le formulaire mono-critère.

La clé `absolute_object` est obligatoire. Elle permet de spécifier à openMairie le nom du modèle l'objet recherché. Ce nom est celui du fichier dans `obj/`, ici `om_utilisateur.class.php` (sans son extension).

4.6.5.3.2.2 Autres paramètres

Wildcard

Le wildcard permet de rendre la recherche stricte ou approximative.

Cette option peut se configurer pour un ou plusieurs modèles particuliers dans les fichiers correspondants du répertoire `sql/` de l'application. Elle peut également être configurée de manière globale pour l'ensemble dans modèle à partir du fichier `dyn/tab.inc.php`.

Par défaut, il est paramétré de la manière suivante :

```
<?php

$options[] = array('type' => 'wildcard', 'left' => '%', 'right' => '%');

?>
```

- `left` détermine, dans la requête SQL de recherche, le caractère ajouté au début (à gauche) de la valeur recherchée;
- `right` détermine, dans la requête SQL de recherche, le caractère ajouté en fin (à droite) de la valeur recherchée.

Avec cette configuration lorsque le mot « admin » est recherché dans une colonne, toutes les valeurs contenant « admin » sont retournées.

En modifiant la configuration de cette manière :

```
<?php

$options[] = array('type' => 'wildcard', 'left' => '', 'right' => '%');

?>
```

Seules les valeurs **commençant** par « admin » seront retournées.

Enfin avec :

```
<?php

$options[] = array('type' => 'wildcard', 'left' => '', 'right' => '');

?>
```

Seules les valeurs égales **exactement** à « admin » seront retournées.

4.6.5.3.3 Configuration des critères de recherche

La recherche avancée ne fonctionnera pas tant que la liste des champs du formulaire multi-critères n'aura pas été créée. Ces champs sont appelés ici des critères de recherche.

4.6.5.3.3.1 Configuration simple

Un critère de recherche est représenté par un tableau PHP contenant sa configuration.

```
<?php

$champs['identifiant_utilisateur'] =
    array('colonne' => 'om_utilisateur',
          'table' => 'om_utilisateur',
```

(suite sur la page suivante)

(suite de la page précédente)

```

        'type' => 'text',
        'libelle' => _('Identifiant'),
        'taille' => 10,
        'max' => 8));

?>

```

La clé `identifiant_utilisateur` est le nom du champ HTML qui sera affiché sur le formulaire.

La clé `colonne` est obligatoire. Elle contient le nom de la colonne de la base de données qui sera interrogée si la variable `$_POST` contient la clé `identifiant_utilisateur`.

La clé `table` est obligatoire. Elle contient le nom de la table de la base de données qui sera interrogée si la variable `$_POST` contient la clé `identifiant_utilisateur`.

La clé `'type'` est obligatoire. Elle contient le type du champ HTML à afficher. Cela peut être `date`, `text`, `select`, ou tout autre méthode de la classe `formulaire`. Pour les champs de type `select`, le nom du champ HTML doit être le même que le nom de la colonne.

La clé `libelle` est obligatoire. Elle contient le libellé qui sera affiché à côté du champ dans le formulaire de recherche.

La clé `taille` est optionnelle. Elle contient la taille du champ HTML (attribut HTML `size`).

La clé `max` est optionnelle. Elle contient la longueur maximale de la valeur du champ HTML (attribut HTML `maxlength`).

Une fois tous les critères de recherche configurés, il faudra simplement vérifier que le tableau des critères est bien utilisé par l'option de type `search`.

Exemple de formulaire pour le tableau du modèle `om_utilisateur`:

```

<?php

$champs = array();

$champs['login'] = array(
    'table' => 'om_utilisateur',
    'colonne' => 'login',
    'type' => 'text',
    'libelle' => _('Login'));

$champs['email'] = array(
    'table' => 'om_utilisateur',
    'colonne' => 'email',
    'type' => 'text',
    'libelle' => _('E-mail'));

$champs['om_profil'] = array(
    'table' => 'om_utilisateur',
    'colonne' => 'om_profil',
    'type' => 'select',
    'libelle' => _('Profil'));

$options[] = array('type' => 'search',
    'display' => true,
    'advanced' => $champs,
    'default_form' => 'advanced',
    'absolute_object' => 'om_utilisateur');

```

(suite sur la page suivante)

```
?>
```

4.6.5.3.3.2 Configuration avancée

4.6.5.3.3.3 Créer un intervalle de date

Exemple : recherche des utilisateurs créés entre telle et telle date.

```
<?php

$champs['date_de_creation'] =
    array('colonne' => 'creation_date',
          'table' => 'user',
          'libelle' => _('Date de creation'),
          'type' => 'date',
          'where' => 'intervaldate');

?>
```

Cette configuration permet de créer deux champs HTML datepicker :

- date_de_creation_min : permettra de saisir une date minimale
- date_de_creation_max : permettra de saisir une date maximale

Ces champs permettent de rechercher les utilisateurs dont la date de créations est incluse dans l'intervalle saisi, bornes comprises. Il est possible de ne saisir qu'une seule date afin de rechercher les utilisateurs ayant été créés avant ou après une date particulière.

4.6.5.3.3.4 Créer un champ de recherche avec menu déroulant personnalisé

Exemple : recherche des utilisateurs administrateurs.

Dans cet exemple, l'information se trouve directement dans la table interrogée.

```
<?php

// soit 'user' une table contenant une colonne 'is_admin'

$args = array();
$args[0] = array('', 'true', 'false');
$args[1] = array(_('Tous'), _('Oui'), _('Non'));

$champs['administrator'] =
    array('colonne' => 'is_admin',
          'table' => 'user',
          'libelle' => _('Administrateur'),
          'type' => 'select',
          'subtype' => 'manualselect',
          'args' => $args);

?>
```

Cette configuration permet de créer un champ HTML de type select avec trois choix :

- Tous (valeur "");

- Oui (valeur `true`);
- Non (valeur `false`).

Le tableau `$args[0]` contient les valeurs associées aux choix. Elles seront recherchées telles quelles dans la base de données.

En sélectionnant « Oui », la requête SQL de recherche sera construite comme suit :

```
-- PostgreSQL
WHERE user.is_admin::varchar like 'true'
```

Il est possible de saisir n'importe quelle chaîne de caractères dans `$args[0]` et pas seulement des valeurs booléennes.

Attention : Cette recherche n'est pas sensible à la casse. Plusieurs fonctions de formatage sont appelées sur `user.is_admin` avant de tester l'égalité.

4.6.5.3.3.5 Tester si une donnée est présente ou non dans un groupe de données

Exemple : recherche des utilisateurs administrateurs.

Dans cet exemple, l'information se trouve non pas dans la table utilisateur mais dans la table administrateur disposant d'une colonne `user_id` (clé étrangère). Il nous faut utiliser une sous-requête pour récupérer l'ensemble des identifiants de la table administrateur afin de tester si un identifiant utilisateur est effectivement présent dans cette liste.

```
<?php

// soit 'user' une table contenant pas la colonne 'is_admin'
// soit 'admin' une table contenant une colonne 'user_id'

$args = array();
$args[0] = array('', 'true', 'false');
$args[1] = array(_('Tous'),
                 _('Administrateurs'),
                 _('Utilisateurs simples'));

$subquery = 'SELECT user_id FROM admin';

$champs['administrator'] =
    array('colonne' => 'id',
          'table' => 'user',
          'libelle' => _('Administrateur'),
          'type' => 'select',
          'subtype' => 'manualselect',
          'where' => 'insubquery',
          'args' => $args,
          'subquery' => $subquery);

?>
```

Cette configuration permet de créer un champ HTML de type `select` avec trois choix :

- Tous (valeur `""`);
- Administrateurs (valeur `true`);
- Utilisateurs simples (valeur `false`).

Le tableau `$args[0]` contient les valeurs associées aux choix. La valeur `true` indique que les identifiants des utilisateurs doivent se trouver dans la sous-requête. La valeur `false` indique qu'ils ne doivent pas se trouver dans

la sous-requête. Contrairement à l'exemple « Créer un champ de recherche avec menu déroulant personnalisé », les valeurs ne seront pas recherchées telles quelles dans la base de données et ne doivent surtout pas être modifiées.

En sélectionnant « Administrateurs », la requête SQL de recherche sera construite comme suit :

```
WHERE user.id IN (SELECT user_id FROM admin)
```

4.6.6 Le composant openMairie

tab.php utilise les méthodes d'om_table.class.php qui est une classe d'openMairie

```
core/om_table.class.php
```

Les méthodes de ce composant peuvent être surchargées dans obj/om_table.class.php

4.7 Les formulaires

Sommaire

- *Les formulaires*
 - *Introduction*
 - *Consultation*
 - *Ajout*
 - *Modification*
 - *Suppression*
 - *Accès*
 - *Les scripts scr/form.php et scr/sousform.php*
 - *Configuration via le script sql/pgsql/<OBJ>.inc.php*
 - *\$ent*
 - *\$sousformulaire*
 - *\$sousformulaire_parameters*
 - *Configuration via le script sql/pgsql/<OBJ>.form.inc.php*
 - *\$form_title*
 - *Les fonctions*
 - *La fonction verrou*
 - *La fonction directlink*
 - *Actions du menu contextuel de la consultation*
 - *Définition des actions dans les attributs de la classe de l'objet*
 - *CRUD*
 - *Définition des actions dans *.form.inc.php (obsolète)*
 - *Description de la classe dbform*
 - *Présentation des méthodes de la classe*
 - *Méthodes d'initialisation de l'affichage du formulaire*
 - *Méthodes d'actions*
 - *Gestion des transactions lors de l'appel aux méthodes d'actions*
 - *Méthodes appelées lors de la validation*
 - *Méthodes permettant d'afficher des informations spécifiques.*
 - *Description de la classe formulaire*
 - *Méthodes d'affichage de widgets*
 - *Les méthodes de construction et d'affichage*

- Les méthodes *assesseurs* changent les valeurs des attributs de l'objet formulaire
- Custom de l'application

4.7.1 Introduction

Les formulaires openMairie sont une visualisation d'un objet d'une classe métier. Les formulaires permettent la consultation, l'ajout, la modification et la suppression d'enregistrements des tables de la base de données.

4.7.1.1 Consultation

La consultation d'un élément est construite de la même façon qu'un formulaire. Elle contient une liste d'actions contextuelles configurable. Les données ne sont pas éditables.

The screenshot displays the openMairie web interface. At the top, the header includes the 'OpenMAIRIE' logo, the user 'admin', the location 'Ville d'ARLES', a password field, and a 'Ta' (Tableau de bord) link. A left sidebar contains a navigation menu with categories like 'Export', 'Paramétrage', 'Administration', 'Gestion Des Utilisateurs', 'Tableaux De Bord', 'Sig', and 'Options Avancees'. The 'Utilisateur' option is highlighted. The main content area shows a breadcrumb trail: 'Administration > Utilisateur > 1 ADMINISTRATEUR'. Below this, a tab labeled 'Utilisateur' is active, displaying a form with the following details: Utilisateur 1, nom Administrateur, email contact@openmairie.org, login admin, mot de passe ***** (masked), and Profil ADMINISTRATEUR. A blue 'Retour' button is at the bottom left of the form. On the right, a red-bordered box highlights a set of action icons: a pencil for 'Mo' (Modifier), a red 'X' for 'Sup' (Supprimer), and a document icon for 'Édi' (Éditer). Below these icons, the text 'Actions c' is partially visible. At the bottom of the page, a footer bar contains the text 'openExemple Version 4.3.0-dev | Documentation | openMairie.org'.

4.7.1.2 Ajout

L'ajout permet l'édérations de données. Lors de la validation, un traitement spécifique des données est effectué. Si la clé primaire de la table est automatique alors elle est générée.

4.7.1.3 Modification

L'ouverture d'un élément en modification permet l'édérations de données déjà existantes, lors de la validation du formulaire les données sont traitées, vérifiées puis envoyées dans la base.

The screenshot shows the OpenMairie administration interface. The top header includes the OpenMAIRIE logo, the user 'admin', the location 'Ville d'ARLES', and a password field. A left sidebar contains a navigation menu with options like 'Export', 'Paramétrage', 'Administration', 'Collectivité', 'Paramètre', 'Gestion Des Utilisateurs', 'Profil', 'Droit', 'Utilisateur', 'Tableaux De Bord', 'Widget', 'Sig', 'Om_sig_map', 'Om_sig_wms', 'Options Avancées', 'Import', and 'Générateur'. The main content area is titled 'Administration ➔ Utilisateur ➔ 1' and shows a form for editing a user. The form fields are: 'Utilisateur *' with value '1', 'nom *' with value 'Administrateur', 'email *' with value 'contact@openmairie.org', 'login *' with value 'admin', 'mot de passe *' with masked characters '*****', and 'Profil *' with a dropdown menu showing 'ADMINISTRATEUR'. Below the form is a button 'Modifier l'enregistrement de la table : 'Utilisateur'' and a 'Retour' link. At the bottom right, there is a footer bar with 'openExemple Version 4.3.0-dev | Documentation | openMairie.org'.

4.7.1.4 Suppression

Accessible depuis la liste des actions contextuelles, une confirmation est demandée pour chaque suppression.

4.7.1.5 Accès

L'accès aux formulaires se fait depuis un *tableau d'éléments* ou depuis la consultation d'un élément via le menu contextuel.

Par défaut, depuis les tableaux, les actions d'ajout et consultation sont disponible.

4.7.2 Les scripts *scr/form.php* et *scr/sousform.php*

Ces scripts sont appelés pour afficher un formulaire. Ilsinstancient l'objet et appellent la méthode formulaire de celui-ci.

La gestion des formulaires se base sur deux classes :

- formulaire : core/om_formulaire.class.php
- dbform : core/om_dbform.class.php

La classe « formulaire » permet la gestion de l'affichage et « dbform » gère le traitement des données et la liaison à la base de données.

Ces scripts prennent plusieurs paramètres :

- obj : nom de la classe pour laquelle on souhaite afficher le formulaire
- action : type d'action (ajout, modification, suppression, consultation)
- idx : identifiant (dans la base de données) de l'élément sur lequel on souhaite effectuer l'action
- retour : deux valeurs possible tab ou form selon l'origine de l'action

Le paramètre « action » peut prendre 4 valeurs :

- 0 : affiche un formulaire d'ajout, le paramètre idx n'est donc pas nécessaire.
- 1 : affiche le formulaire de modification.
- 2 : affiche le formulaire de suppression.
- 3 : affiche le formulaire de consultation.

Les autres paramètres passés permettent de conserver la configuration du tableau d'origine.

4.7.3 Configuration via le script `sql/pgsql/<OBJ>.inc.php`

4.7.3.1 `$ent`

Titre (fil d'ariane) de la page.

```
<?php
$ent = _("administration")." -> "._("om_parametre");
?>
```

Ces variables peuvent être accessibles par les classes métier qui peuvent modifier le fil d'ariane depuis un formulaire ou un sous formulaire

```
function getFormTitle($ent) {
    return $ent;
}

function getSubFormTitle($subEnt) {
    return $subEnt;
}
```

4.7.3.2 `$sousformulaire`

Liste des onglets (autre que le principal).

```
<?php
$sousformulaire = array(
    "consultation",
    "instruction",
);
?>
```

4.7.3.3 `$sousformulaire_parameters`

Configuration spécifique des onglets (autre que le principal).

```
<?php
$sousformulaire_parameters = array(
    "consultation" => array(
        "title" => _("CAP(s)"),
        "href" => "../scr/sousform.php?obj=consultation_specific&action=12&idx=0",
    ),
    "instruction" => array(
        "title" => _("Unité(s) orga."),
    ),
);
?>
```

4.7.4 Configuration via le script `sql/pgsql/<OBJ>.form.inc.php`

4.7.4.1 `$form_title`

Titre de l'onglet principal du formulaire.

```
<?php
$form_title = "Organigramme";
?>
```

4.7.5 Les fonctions

4.7.5.1 La fonction `verrou`

Note : Cette description correspond au fonctionnement du verrou depuis la version 4.5.0.

La fonction **verrou** a pour objectif d'empêcher la double soumission de formulaire côté serveur. Elle est active dans les `VIEW formulaire()` et `sousformulaire()`. A chaque affichage de formulaire, lorsqu'un bouton est affiché alors on insère un champ caché (input de type hidden) qui contient comme valeur un identifiant généré et supposé unique, puis on stocke cet identifiant dans une liste dédiée dans la variable de session de l'utilisateur connecté. Lors de la soumission du formulaire, on vérifie que la valeur de l'identifiant postée avec le formulaire est bien présente dans la liste dédiée dans la variable de session, si c'est le cas on enlève la valeur de cette liste et on exécute le traitement. Si ce n'est pas le cas, cela signifie que le formulaire a déjà été soumis au préalable donc on affiche une erreur à l'utilisateur.

Les trois méthodes de l'ancienne implémentation `verrouille()`, `deverrouille()` et `testverrou()` ont été vidées et conservées pour la rétro-compatibilité des applications. Tous les appels à ces méthodes ont été supprimés du framework. Ces méthodes sont vides dans la version 4.5.0 et seront supprimées dans la 4.6.0.

4.7.5.2 La fonction `directlink`

La fonction **directlink** a pour objectif d'accéder via une URL à une vue spécifique d'un objet dans un onglet dans le contexte d'un formulaire. Ce sont les script `spg/direct_link.php` et `scr/form.php` qui sont en charge de réaliser l'opération.

Paramètres du script `spg/direct_link.php` qui récupère l'identifiant de l'objet parent lié et l'identifiant de l'onglet correspondant à la classe de l'objet à afficher :

- `obj` (obligatoire) : classe de l'objet contexte
- `action` (obligatoire) : action sur l'objet contexte
- `idx` (optionnel soit `idx` soit `direct_field`) : identifiant de l'objet contexte
- `direct_field` (optionnel soit `idx` soit `direct_field`) : nom du champ contenant l'identifiant de l'objet contexte
- `direct_form` (obligatoire) : nom de l'objet direct à afficher
- `direct_action` (obligatoire) : action à effectuer sur l'objet direct
- `direct_idx` (obligatoire) : identifiant de l'objet direct à afficher

Paramètres du script `scr/form.php` :

- `obj`
- `action`
- `idx`
- `direct_form`
- `direct_action`
- `direct_idx`
- identifiant de l'onglet `#ui-tabs`

Par exemple, pour accéder au formulaire de modification de l'utilisateur dont l'identifiant est le 1 dans le contexte de sa collectivité directement via une URL voici l'URL à appeler :

```
../spg/direct_link.php?obj=om_collectivite&action=3&direct_field=om_collectivite&
↪direct_form=om_utilisateur&direct_action=1&direct_idx=1
```

Celle ci va rediriger vers :

```
../scr/form.php?obj=om_collectivite&action=3&idx=1&direct_form=om_utilisateur&direct_
↪idx=1&direct_action=1#ui-tabs-1
```

Note : *Limitations* - Ne peut fonctionner que si :

- la vue par défaut de l'onglet est un soustab standard et non une vue par défaut
- l'objet doit contenir dans son modèle de données un champ contenant l'identifiant de l'objet du contexte souhaité si on utilise le paramètre `direct_field`

4.7.6 Actions du menu contextuel de la consultation

Dans `dyn/config.inc.php` :

```
<?php
/**
 * Parametre de gestion des nouvelles actions
 * Permet de definir si la gestion des actions se fait dans la classe ou non.
 * Si on decide d'utiliser les nouvelles actions alors il n'y a pas de
 * retro-compatibilité, les actions supplémentaires de portlet initialement
 * déclarées dans sql/pgsql/*.inc.php ne fonctionneront plus et devront
 * être initialisées dans les attributs de la classe ciblée.
 * Default : $config['activate_class_action'] = true;
 */
$config['activate_class_action'] = true;
?>
```

4.7.6.1 Définition des actions dans les attributs de la classe de l'objet

La configuration se fait dans les attributs des classes (`obj/*.class.php`).

L'ajout d'une action se présente de cette façon :

```
<?php
function init_class_actions() {
    // On récupère les actions génériques définies dans la méthode
    // d'initialisation de la classe parente
    parent::init_class_actions();

    // ACTION - 002 - supprimer
    //
    $this->class_actions[2] = array(
        "portlet" => array(
            "libelle" => "supprimer",
            "class" => "delete-16",
            "order" => 20,
            "description" => _("Accéder au formulaire de suppression de l
↪'enregistrement"),
```

(suite sur la page suivante)

(suite de la page précédente)

```

    ),
    "method" => "supprimer",
    "button" => "supprimer",
    "permission_suffix" => "supprimer",
    "condition" => "delete_coll_condition"
  );
}
?>

```

La clé du tableau correspond à la valeur \$maj, le paramètre « method » correspond à la méthode appelée lors de la validation du formulaire, « button » est le texte du bouton de validation, « permission_suffix » est le suffixe du droit qui sera testé lors de l’affichage de l’action, « condition » permet de définir une méthode qui sera appelée avant l’affichage de l’action dans le portlet, si cette méthode retourne « true » l’action sera affichée.

Si la clé « portlet » est définie l’action correspondante sera affichée (sous condition), la clé « libelle » est le texte affiché sur le lien, la classe définie dans « class » sera ajoutée à celles du lien, « order » permet de définir l’ordre, la clé « url » peut être utilisée pour définir une url spécifique.

Les actions de classes permettent de surcharger les actions ajouter, modifier, consulter et supprimer définies dans core/om_db_form.class.php.

L’action qui porte le numéro 999 est réservée à la recherche avancée.

4.7.6.2 CRUD

Les formulaires de base sont facilement reproductibles : il existe un mode pour chaque action : Create, Read, Update et Delete.

En définissant le paramètre « crud » adéquat, vous aurez automatiquement la vue et sa méthode de traitement sans développement supplémentaire.

Ainsi cette action « ajouter_bis » est une copie fonctionnelle et suffisante de l’action ajouter :

```

<?php
// ACTION - 004 - ajouter_bis
//
$this->class_actions[4] = array(
    "identifiant" => "ajouter_bis",
    "permission_suffix" => "ajouter",
    "crud" => "create",
);
?>

```

4.7.6.3 Définition des actions dans *.form.inc.php (obsolète)

Dans dyn/config.inc.php :

```

<?php
/**
 * Parametre de gestion des nouvelles actions
 * Permet de definir si la gestion des actions se fait dans la classe ou non.
 * Si on decide d'utiliser les nouvelles actions alors il n'y a pas de
 * retro-compatibilité, les actions supplémentaires de portlet initialement
 * déclarées dans sql/pgsql/*.inc.php ne fonctionneront plus et devront
 * être initialisées dans les attributs de la classe ciblée.
 */

```

(suite sur la page suivante)

(suite de la page précédente)

```

* Default : $config['activate_class_action'] = true;
*/
$config['activate_class_action'] = false;
?>

```

La configuration des actions du menu contextuel des formulaires en consultation se fait via les scripts `sql/sqbd/objet.form.inc.php`

Dans ces scripts, peuvent être surchargés, la liste des champs (ordre ou champs affichés), requêtes sql permettant de remplir les widget de formulaires ainsi que les actions du menu contextuel.

L'ajout d'une action se présente de cette façon :

```

<?php
$portlet_actions['edition'] = array(
    'lien' => '../pdf/pdflettrettype.php?obj=om_utilisateur&idx=',
    'id' => '',
    'lib' => '<span class="om-prev-icon om-icon-16 om-icon-fix pdf-16">'.__('Edition').
    '</span>',
    'ajax' => false,
    'ordre' => 21,
    'description' => _("Télécharger le courrier de l'utilisateur au format PDF"),
);
?>

```

4.7.7 Description de la classe dbform

class dbform (\$id, &\$db, \$DEBUG = false)

Cette classe est centrale dans l'application. Elle est la classe parente de chaque objet métier. Elle comprend des méthodes de gestion (initialisation, traitement, vérification, trigger) des valeurs du formulaire. Elle fait le lien entre la base de données et le formulaire. Elle contient les actions possibles sur les objets (ajout, modification, suppression, consultation).

4.7.7.1 Présentation des méthodes de la classe

Les méthodes de dbform peuvent être surchargées dans `obj/om_dbform.class.php` ainsi que dans toutes les classes métier.

4.7.7.2 Méthodes d'initialisation de l'affichage du formulaire

dbform.formulaire (\$enteteTab, \$validation, \$maj, &\$db, \$postVar, \$aff, \$DEBUG = false, \$idx, \$premier = 0, \$recherche = "", \$tricol = "", \$idz = "", \$selection-col = "", \$advs_id = "", \$valide = "", \$retour = "", \$actions = array(), \$extra_parameters = array())

Méthode d'initialisation de l'affichage de formulaire.

dbform.sousformulaire (\$enteteTab, \$validation, \$maj, &\$db, \$postVar, \$premiersf, \$DEBUG, \$idx, \$idxformulaire, \$retourformulaire, \$typeformulaire, \$objsf, \$tricolsf, \$retour = "", \$actions = array())

Méthode d'initialisation de l'affichage de sous formulaire.

Ces méthodesinstancient un objet « formulaire » et initialisent certains de ses attributs via les méthodes suivantes :

dbform.setVal (&\$form, \$maj, \$validation)

Permet de définir les valeurs des champs en contexte formulaire

`dbform.setValsousformulaire (&$form, $maj, $validation, $idxformulaire, $retourformulaire, $typeformulaire)`
Permet de définir les valeurs des champs en contexte sous-formulaire

`dbform.set_form_default_values (&$form, $maj, $validation)`
Permet de définir les valeurs des champs en contextes formulaire et sous-formulaire

`dbform.setType (&$form, $maj)`
Permet de définir le type des champs

`dbform.setLib (&$form, $maj)`
Permet de définir le libellé des champs

`dbform.setTaille (&$form, $maj)`
Permet de définir la taille des champs

`dbform.setMax (&$form, $maj)`
Permet de définir le nombre de caractères maximum des champs

`dbform.setSelect (&$form, $maj, $db, $DEBUG = false)`
Méthode qui effectue les requêtes de configuration des champs

`dbform.init_select (&$form = null, &$db = null, $maj, $debug, $field, $sql, $sql_by_id, $om_validite = false, $multiple = false)`
Méthode qui permet la configuration des select et select multiple, elle effectue les requêtes et met en forme le tableau des valeurs à afficher. Il est possible de définir si le champ lié est affecté par une date de validité ou de configurer l’affichage de select_multiple.

`dbform.setOnchange (&$form, $maj)`
Permet de définir l’attribut « onchange » sur chaque champ

`dbform.setOnkeyup (&$form, $maj)`
Permet de définir l’attribut « onkeyup » sur chaque champ

`dbform.setOnclick (&$form, $maj)`
Permet de définir l’attribut « onclick » sur chaque champ

`dbform.setGroupe (&$form, $maj)`
Permet d’aligner plusieurs champs (obsolète depuis la version 4.3.0)

`dbform.setRegroupe (&$form, $maj)`
Permet de regrouper les champs dans des fieldset (obsolète depuis la version 4.3.0)

`dbform.setLayout (&$form, $maj)`
Méthode de mise en page, elle permet de gérer la hiérarchie d’ouverture et fermeture des balises div et fieldset avec les méthodes :

- `formulaire.setBloc ($champ, $contenu, $libelle = "", $style = "")`
permet d’ouvrir/fermer (\$contenu=D/F) une balise div sur un champ (\$champ), avec un libellé (\$libelle) et un attribut class (\$style).
 - une liste de classes css pour fieldset est disponible : “group” permet une mise en ligne des champs contenu dans le div et “col_1 à col_12” permet une mise en page simplifiée (par exemple : « col_1 » permet de définir une taille dynamique de 1/12ème de la page , col_6 correspond à 6/12 soit 50% de l’espace disponible).
 - il est possible de créer et ajouter des classes css aux différents div afin d’obtenir une mise en page personnalisé.
- `formulaire.setFieldset ($champ, $contenu, $libelle = "", $style = "")`
permet d’ouvrir/fermer (\$contenu=D/F) un fieldset sur un champ (\$champ), avec une légende (\$libelle) et un attribut class (\$style).
 - une liste de classes css pour fieldset est disponible : “collapsible” ajoute un bouton sur la légende (jQuery) afin de refermer le fieldset et “startClosed” idem à la différence que le fieldset est fermé au chargement de la page.
- exemple d’implémentation de la méthode setLayout() sans utiliser les méthodes setGroupe() et setRegroupe() :

```

<?php
function setLayout(&$form, $maj) {
    //Ouverture d'un div sur une colonne de 1/2 (6/12) de la_
    ↪ largeur du
    //conteneur parent
    $form->setBloc('om_collectivite', 'D', "", "col_6");
    //Ouverture d'un fieldset
    $form->setFieldset('om_collectivite', 'D', _('om_collectivite
    ↪ '),
        "collapsible");
    //Ouverture d'un div les champs compris entre
    //"om_collectivite" et "actif"
    //la classe group permet d'afficher les champs en ligne
    $form->setBloc('om_collectivite', 'D', "", "group");
    //Fermeture du groupe
    $form->setBloc('actif', 'F');
    //Fermeture du fieldset
    $form->setFieldset('actif', 'F', '');
    //Fermeture du div de 50%
    $form->setBloc('actif', 'F');

    //Ouverture d'un div sur une colonne de 1/2 de la largeur du
    //conteneur parent
    $form->setBloc('orientation', 'D', "", "col_6");
    $form->setFieldset('orientation', 'D',
        _("Parametres generaux du document"),
        "startClosed");
    $form->setBloc('orientation', 'D', "", "group");
    $form->setBloc('format', 'F');

    $form->setBloc('footerfont', 'D', "", "group");
    $form->setBloc('footertaille', 'F');

    $form->setBloc('logo', 'D', "", "group");
    $form->setBloc('logotop', 'F');
    $form->setFieldset('logotop', 'F', '');
    $form->setBloc('logotop', 'F');

    //Ouverture d'un div de largeur maximum sur un seul champ
    $form->setBloc('titre', 'DF', "", "col_12");

    //Ouverture d'un div de largeur maximum
    $form->setBloc('titreleft', 'D', "", "col_12");
    $form->setFieldset('titreleft', 'D',
        _("Parametres du titre du document"),
        "startClosed");
    $form->setBloc('titreleft', 'D', "", "group");
    $form->setBloc('titrehauteur', 'F');

    $form->setBloc('titrefont', 'D', "", "group");
    $form->setBloc('titrealign', 'F');
    $form->setFieldset('titrealign', 'F', '');
    $form->setBloc('titrealign', 'F');

    //Ouverture d'un div de largeur maximum sur un seul champ
    $form->setBloc('corps', 'DF', "", "col_12");

```

(suite sur la page suivante)

(suite de la page précédente)

```

//Ouverture d'un div de largeur maximum
$form->setBloc('corpsleft', 'D', "", "col_12");
    $form->setFieldset('corpsleft', 'D',
        _("Parametres du corps du document"),
        "startClosed");
    $form->setBloc('corpsleft', 'D', "", "group");
    $form->setBloc('corpshauteur', 'F');

    $form->setBloc('corpsfont', 'D', "", "group");
    $form->setBloc('corpsalign', 'F');
    $form->setFieldset('corpsalign', 'F', '');
    $form->setBloc('corpsalign', 'F');

//Ouverture d'un div de largeur maximum sur un seul champ
$form->setBloc('om_sql', 'DF', "", "col_12");

//Ouverture d'un div de 1/2 de la largeur du conteneur parent
$form->setBloc('om_sousetat', 'D', "", "col_6");
    $form->setFieldset('om_sousetat', 'D',
        _("Sous etat(s) : selection"),
        "startClosed");
    $form->setBloc('om_sousetat', 'D', "", "group");
    $form->setBloc('sousetat', 'F');
    $form->setFieldset('sousetat', 'F', '');
    $form->setBloc('sousetat', 'F');

//Ouverture d'un div de 1/2 de la largeur du conteneur parent
$form->setBloc('se_font', 'D', "", "col_6");
    $form->setFieldset('se_font', 'D',
        _("Sous etat(s) : police / marges / ↪
↪couleur"),
        "startClosed");
    $form->setBloc('se_font', 'D', "", "group");
    $form->setBloc('se_couleurtexte', 'F');
    $form->setFieldset('se_couleurtexte', 'F', '');
    $form->setBloc('se_couleurtexte', 'F');
}
?>

```

4.7.7.3 Méthodes d'actions

Ces méthodes sont appelées lors de la validation du formulaire.

`dbform.ajouter ($val, &$db = NULL, $DEBUG = false)`

Cette méthode permet l'insertion de données dans la base, elle appelle toutes les méthodes de traitement, vérification et méthodes spécifiques à l'ajout.

`dbform.modifier ($val = array(), &$db = NULL, $DEBUG = false)`

Cette méthode permet la modification de données dans la base, elle appelle toutes les méthodes de traitement et vérification des données retournées par le formulaire.

`dbform.supprimer ($val = array(), &$db = NULL, $DEBUG = false)`

Cette méthode permet la suppression de données dans la base, elle appelle toutes les méthodes de traitement et vérification des données retournées par le formulaire.

4.7.7.4 Gestion des transactions lors de l'appel aux méthodes d'actions

Afin de vérifier les erreurs de base de données, la méthode `isError` est appelée, si la valeur `true` lui est passée en second paramètre elle ne stop pas l'exécution mais retour `true` ou `false`. Cela dans le but d'appeler ces méthodes sur des objets métier instanciés manuellement dans des contextes qui n'utilise pas la classe formulaire. Exemple : lors de la création d'un web service qui instancierait une classe, si une erreur de base de données se produit, le script s'arrête et aucun message ne peut être transmis au client du web service, ce qui ne se produit pas si le second paramètre est défini à `true`.

Il est important d'instancier un objet métier et d'appeler les méthodes ajouter, modifier ou supprimer pour effectuer un changement sur celui-ci car toutes les méthodes de trigger seront appelées.

4.7.7.5 Méthodes appelées lors de la validation

`dbform.setValFAjout ($val = array())`

Méthode de traitement des données retournées par le formulaire (utilisé lors de l'ajout)

`dbform.setvalF ($val = array())`

Méthode de traitement des données retournées par le formulaire

`dbform.verifier ($val = array(), &$db = NULL, $DEBUG = false)`

Méthode de vérification des données et de retour d'erreurs

`dbform.verifierAjout ($val = array(), &$db = NULL)`

Méthode de vérification des données et de retour d'erreurs (utilisé lors de l'ajout)

`dbform.setId (&$db = NULL)`

Initialisation de la clé primaire (si clé automatique lors de l'ajout)

`dbform.cleSecondaire ($id, &$db = NULL, $val = array(), $DEBUG = false)`

Cette méthode est appelée lors de la suppression d'un objet, elle permet de vérifier si l'objet supprimé n'est pas lié à une autre table pour en empêcher la suppression.

`dbform.triggerajouter ($id, &$db = NULL, $val = array(), $DEBUG = false)`

Permet d'effectuer des actions avant l'insertion des données dans la base

`dbform.triggerajouterapres ($id, &$db = NULL, $val = array(), $DEBUG = false)`

Permet d'effectuer des actions après l'insertion des données dans la base

`dbform.triggermodifier ($id, &$db = NULL, $val = array(), $DEBUG = false)`

Permet d'effectuer des actions avant la modification des données dans la base

`dbform.triggermodifierapres ($id, &$db = NULL, $val = array(), $DEBUG = false)`

Permet d'effectuer des actions après la modification des données dans la base

`dbform.triggersupprimer ($id, &$db = NULL, $val = array(), $DEBUG = false)`

Permet d'effectuer des actions avant la modification des données dans la base

`dbform.triggersupprimerapres ($id, &$db = NULL, $val = array(), $DEBUG = false)`

Permet d'effectuer des actions après la modification des données dans la base

4.7.7.6 Méthodes permettant d'afficher des informations spécifiques.

Ces méthodes fournissent des points d'entrée dans les formulaires et les sous formulaires (voir opencourrier : liaison de courrier dans `obj/courrier.class.php`)

`formSpecificContent ($maj)`

Cette méthode à surcharger permet d'afficher des informations spécifiques en fin de formulaire.

`sousFormSpecificContent ($maj)`

Cette méthode à surcharger permet d'afficher des informations spécifiques en fin de sous formulaire.

afterFormSpecificContent ()

Cette méthode à surcharger permet d'afficher des informations spécifiques après le formulaire.

afterSousFormSpecificContent () { }

Cette méthode à surcharger permet d'afficher des informations spécifiques après le sous formulaire.

4.7.8 Description de la classe formulaire

```
class formulaire ($unused = NULL, $validation, $maj, $champs = array(), $val = array(), $max = array())
```

Cette classe permet une gestion complète de l'affichage d'un formulaire.

Les méthodes de core/om_formulaire.class.php peuvent être surchargées dans obj/om_formulaire.class.php

4.7.8.1 Méthodes d'affichage de widgets

Les widgets sont des éléments de formulaire, ils sont composés d'un ou plusieurs champs. Chaque méthode permet d'afficher un seul widget.

```
formulaire.text()
```

champ texte (format standard)

```
formulaire.hidden()
```

champ non visible avec valeur conservée

```
formulaire.password()
```

champ password

```
formulaire.textdisabled()
```

champ texte non modifiable (grisé)

```
formulaire.textreadonly()
```

champ texte non modifiable

```
formulaire.hiddenstatic()
```

champ non modifiable, la valeur est récupéré par le formulaire.

```
formulaire.hiddenstaticnum()
```

champ numérique non modifiable et valeur récupérer

```
formulaire.static()
```

Valeur affichée et non modifiable

```
formulaire.affichepdf()
```

récupère un nom d'objet (un scan pdf)

```
formulaire.checkbox()
```

case à cocher valeurs possibles : True ou False

```
formulaire.checkboxstatic()
```

affiche Oui/Non, non modifiable (mode consultation)

```
formulaire.checkboxnum()
```

cochée = 1, non cochée = 0

```
formulaire.http()
```

lien http avec target = _blank (affichage dans une autre fenêtre)

```
formulaire.httpclick()
```

lien avec affichage dans la même fenêtre.

```
formulaire.date()
```

date modifiable avec affichage de calendrier jquery

```
formulaire.date2()
```

date modifiable avec affichage de calendrier jquery pour les sous-formulaires

```
formulaire.hiddenstaticdate()
```

date non modifiable Valeur récupéré par le formulaire

`formulaire.datestatic()`
 affiche la date formatée, non modifiable (mode consultation)

`formulaire.textarea()`
 affichage d un textarea

`formulaire.textareamulti()`
 textarea qui récupère plusieurs valeurs d'un select

`formulaire.textareahiddenstatic()`
 affichage non modifiable d'un textarea et récupération de la valeur

`formulaire.pagehtml()`
 affichage d'un textarea et transforme les retours charriot en `</ br>`

`formulaire.select()`
 champ select

`formulaire.selectdisabled()`
 champ select non modifiable

`formulaire.selectstatic()`
 affiche la valeur de la table liée, non modifiable (mode consultation)

`formulaire.selecthiddenstatic()`
 affiche la valeur de la table liée, non modifiable ainsi que la valeur dans un champ hidden

`formulaire.select_multiple()`
 affiche un select multiple, les valeurs passées au formulaires doivent être séparées par une virgule.

`formulaire.select_multiple_static()`
 affiche seulement les valeurs d'un select multiple, les valeurs passées au formulaires doivent être séparées par une virgule.

`formulaire.comboG()`
 permet d'effectuer une corrélation entre un groupe de champ et un identifiant dans les formulaires

`formulaire.comboG2()`
 permet d'effectuer une corrélation entre un groupe de champ et un identifiant dans les sous formulaires

`formulaire.comboD()`
 permet d'effectuer une corrélation entre un groupe de champ et un identifiant dans les formulaires

`formulaire.comboD2()`
 permet d'effectuer une corrélation entre un groupe de champ et un identifiant dans les sous formulaires

`formulaire.upload()`
 fait appel à `spg/upload.php` pour télécharger un fichier

`formulaire.upload2()`
 fait appel à `spg/upload.php` pour télécharger un fichier dans un sous formulaire

`formulaire.voir()`
 fait appel à `spg/voir.php` pour visualiser un fichier

`formulaire.voir2()`
 fait appel à `spg/voir.php` pour visualiser un fichier depuis un sous formulaire

`formulaire.localisation()`
 fait appel à `spg/localisation.php`

`formulaire.localisation2()`
 fait appel à `spg/localisation.php`

`formulaire.rvb()`
 fait appel à `spg/rvb.php` pour affichage de la palette couleur

`formulaire.rvb2()`
 fait appel à `spg/rvb.php` pour affichage de la palette couleur

`formulaire.geom()`
 ouvre une fenêtre `tab_sig.php` pour visualiser ou saisir une géométrie (selon l'action) la carte est définie en `setSelect`

Les widgets comboG, comboD, date, upload, voir et localisation sont à mettre dans les formulaires. Les contrôle comboG2, comboD2, date2, upload2, voir2 et localisation sont à mettre dans les sous formulaires.

Les widgets font appel des scripts d'aide à la saisie stockés dans le répertoire /spg, ils sont appelés par js/script.js. Ce script peut être surchargé dans app/js/script.js.

spg/combo.php

Ce programme est appelé par le champ comboD, comboG, comboD2, comboG2, le paramétrage se fait dans les fichiers :

- dyn/comboparametre.inc.php
- dyn/comboretour.inc.php
- dyn/comboaffichage.inc.php

spg/localisation.php et js/localisation.js

ce programme est liée au champ formulaire « localisation ».

spg/voir.php

Ce script est associé au champ « upload ».

Ce sous programme permet de visualiser un fichier téléchargé sur le serveur (pdf ou image).

spg/upload.php

Ce script utilise la classe core/upload.class.php (composant openMairie).

Le paramétrage des extensions téléchargeables se fait dans dyn/config.inc.php. Le paramétrage de la taille maximale des fichiers téléchargeables se fait dans la classe métier de l'objet.

spg/rvb.php et js/rvb.js

Ce script est associé au champ « rvb » et affiche une palette de couleur pour récupérer un code rvb.

4.7.8.2 Les méthodes de construction et d'affichage

Le formulaire est constitué de div, fieldset et de champs les méthodes suivantes permettent une mise en page structurée.

```
formulaire.entete()  
    ouverture du conteneur du formulaire.  
formulaire.enpied()  
    fermeture du conteneur du formulaire.  
formulaire.afficher()  
    affichage des champs, appelle les méthodes suivante :  
formulaire.debutFieldset()  
    ouverture de fieldset.  
formulaire.finFieldset()  
    fermeture de fieldset  
formulaire.debutBloc()  
    ouverture de div.  
formulaire.finBloc()  
    fermeture de div.  
formulaire.afficherChamp()  
    affichage de champ.
```

4.7.8.3 Les méthodes assesseurs changent les valeurs des attributs de l'objet formulaire

Ces méthodes sont appelées depuis les classes métier, elles permettent la configuration du formulaire.

```
formulaire.setType()
    type de champ
formulaire.setVal()
    valeur du champ
formulaire.setLib()
    libellé du champ
formulaire.setSelect()
    permet de remplir les champs select avec la table liée
formulaire.setTaille()
    taille du champ
formulaire.setMax()
    nombre de caractères maximum acceptés
formulaire.setOnchange()
    permet de définir des actions sur l'événement « onchange »
formulaire.setKeyup()
    permet de définir des actions sur l'événement « onkeyup »
formulaire.setOnClick()
    permet de définir des actions sur l'événement « onclick »
formulaire.setvalF()
    permet de traiter les données avant insert/update dans la base de données
formulaire.setGroupe()
    (obsolète depuis 4.3.0)
formulaire.setRegroupe()
    (obsolète depuis 4.3.0)
formulaire.setBloc($champ, $contenu, $libelle = "", $style = "")
    permet d'ouvrir/fermer ($contenu=D/F/DF) une balise div sur un champ ($champ), avec un libellé ($libelle) et un attribut class ($style).
formulaire.setFieldset($champ, $contenu, $libelle = "", $style = "")
    permet d'ouvrir/fermer ($contenu=D/F/DF) un fieldset sur un champ ($champ), avec une légende ($libelle) et un attribut class ($style).
```

4.7.8.4 Custom de l'application

Il est possible d'ajouter des scripts personnalisés pour les scripts reqmo et import ainsi que pour les classes métier.

Ces scripts peuvent être stockés en dehors de l'application pour des besoins spécifiques qui n'entrent pas dans les champs fonctionnel de base.

Le répertoire où doit être les scripts est à paramétrer dans dyn/include.inc

```
// CUSTOM reqmo - pdf - import
define("PATH_CUSTOM", getcwd()."/../custom/");
```

Il faut ensuite indiquer quels sont les scripts qui surchargent les scripts métiers de l'application dans le fichier dyn/custom.inc.php

```
$custom=array();
$custom['tab']['om_utilisateur'] = '../custom/sql/pgsql/om_utilisateur.inc.php';
$custom['soustab']['om_utilisateur'] = '../custom/sql/pgsql/om_utilisateur.inc.php';
```

(suite sur la page suivante)

(suite de la page précédente)

```
$custom['form']['om_utilisateur'] = '../custom/sql/pgsql/om_utilisateur.form.inc.php';  
$custom['obj']['om_utilisateur'] = '../custom/obj/om_utilisateur.class.php';
```

Enfin il faut créer les surcharges

```
// exemple dans ../custom/sql/pgsql/om_utilisateur.inc.php  
include "../sql/pgsql/om_utilisateur.inc.php";  
$champAffiche = array(  
    'om_utilisateur.om_utilisateur as "'.$_("om_utilisateur").'",  
    'om_utilisateur.nom as "'.$_("nom").'",  
);  
  
// exemple dans ../custom/obj/om_utilisateur.class.php  
require_once "../obj/om_utilisateur.class.php";  
  
class om_utilisateur_custom extends om_utilisateur {  
  
    function om_utilisateur_custom($id,&$db,$debug) {  
        $this->constructeur($id,$db,$debug);  
    } // fin constructeur  
  
    function setType(&$form,$maj) {  
        parent::setType($form, $maj);  
        if ($maj==1)  
            $form->setType("pwd", "hiddenstatic");  
    }  
  
    function setLib(&$form,$maj) {  
        parent::setLib($form, $maj);  
        $form->setLib("nom", "nom dans la classe surchargee");  
    }  
  
}
```

Pour les reqmo et les imports, il n'est pas besoin de paramétrer dyn/custom.inc.php car le framework les récupèrent automatiquement.

Note : Attention, la surcharge du custom ne fonctionne que pour la classe cible et non celles qui éventuellement surcharge dans l'application la classe cible.

4.8 Module “Édition”

Sommaire

- *Module “Édition”*
- *Introduction*
- *Les états et lettres types*
 - *Paramétrer des états*
 - *Paramétrer des lettres-type*
 - *Actif, non actif*

- *Les requêtes*
 - *Description*
 - *SQL*
 - *OBJET*
 - *Modèle de données*
- *Les sous-états*
- *Les champs de fusion*
- *Les variables de remplacement*
 - *Les fichiers de configuration ../dyn/var*pdf.inc*
 - *Les méthodes globales de la classe du fichier ../obj/om_dbform.class.php*
 - *La table de paramètres om_parametre*
- *Les logos*
- *L'éditeur WYSIWYG*
- *Les anciens fichiers de paramétrage*
- *La prévisualisation*
- *Les listings*
 - *Description de la fonctionnalité*
 - *Le fichier de paramétrage ../sql/pgsql/<OBJ>.pdf.inc.php*
- *Les étiquettes*
 - *Description de la fonctionnalité*
 - *Le fichier de paramétrage ../sql/pgsql/<OBJ>.pdfetiquette.inc.php*
- *Composants*

4.8.1 Introduction

Le framework openMairie permet d'effectuer des éditions au format PDF. Ce module est composé de trois éléments fonctionnels :

- les états et lettres types,
- les listings,
- les étiquettes.

4.8.2 Les états et lettres types

C'est la fonctionnalité la plus évoluée du module "Édition", elle permet à l'utilisateur final de composer directement depuis l'interface du logiciel des éditions complexes au format PDF comme des factures, des courriers, des procès verbaux, ... Il est possible d'insérer dans ces éditions : un logo, des sous-états, des champs de fusion.

4.8.2.1 Paramétrer des états

Il est conseillé d'utiliser l'assistant état du générateur.

Les paramètres sont les suivants

```
orientation portrait ou paysage
format="A4", A3
position et nom du logo
titre de l'état
position et caractéristiques du titre
corps de l'état
position et caractéristiques du corps
```

(suite sur la page suivante)

(suite de la page précédente)

```
la requête SQL
les sous-états associés et les caractéristiques
```

Pour le corps et le titre, les zones entre crochets (exemple [nom]) sont les champs de fusion, sélectionnés par la requête SQL.

Ces champs de fusion peuvent être mis en majuscule ou en minuscule selon les besoins grâce aux balises <MAJ></MAJ> et <min></min>.

EX. :

Requête SQL = SELECT nom as nom FROM A ;

Cette phrase dans un PDF “Lorem [nom] dolor sit amet, <MAJ>[nom]</MAJ> adipiscing elit. Curabitur feugiat.” deviendra “Lorem nom dolor sit amet, NOM adipiscing elit. Curabitur feugiat.”

Les variables commençant par « & » sont celles définies dans `dyn/varpdf.inc` (exemple `&aujourd'hui`) et dans la table `om_parametre`.

```
— obj/om_etat.class.php
— core/obj/om_etat.class.php
— gen/obj/om_etat.class.php
— sql/pgsql/om_etat.form.inc.php
— sql/pgsql/om_etat.inc.php
— core/sql/pgsql/om_etat.form.inc.php
— core/sql/pgsql/om_etat.inc.php
— gen/sql/pgsql/om_etat.form.inc.php
— gen/sql/pgsql/om_etat.inc.php
```

4.8.2.2 Paramétrer des lettres-type

Il est conseillé d'utiliser l'assistant lettre-type du générateur.

Les paramètres sont les suivants

```
orientation portrait ou paysage
format="A4", A3
position et nom du logo
titre de la lettre
position et caractéristiques du titre
corps de la lettre
position et caractéristiques du corps
la requête SQL
```

Pour le corps et le titre, les zones entre crochets sont les champs de fusion, sélectionnés par la requête.

Les variables commençant par « & » sont celles définies dans `dyn/varlettretypespdf.inc` (exemple `&aujourd'hui`) et dans la table `om_parametre`.

```
— obj/om_lettrettype.class.php
— core/obj/om_lettrettype.class.php
— gen/obj/om_lettrettype.class.php
— sql/pgsql/om_lettrettype.form.inc.php
— sql/pgsql/om_lettrettype.inc.php
— core/sql/pgsql/om_lettrettype.form.inc.php
— core/sql/pgsql/om_lettrettype.inc.php
— gen/sql/pgsql/om_lettrettype.form.inc.php
— gen/sql/pgsql/om_lettrettype.inc.php
```

4.8.2.3 Actif, non actif

Les sous-états sont liés à un ou plusieurs états.

Les états, sous-états, et lettre type peuvent être « actif » ou « non-actif ».

Par défaut sont pris en compte :

- 1 - l'édition « actif » de la collectivité
- 2 - l'édition « actif » de la multicollectivité
- 3 - l'édition « non-actif » de la multicollectivité

Les éditions d'une collectivité ayant le statut « non-actif » ne sont pas prises en compte.

4.8.2.4 Les requêtes

4.8.2.4.1 Description

Une requête peut être :

- de type SQL
- de type OBJET

4.8.2.4.1.1 SQL

Lorsqu'elle est de type SQL il s'agit d'un ordre *SELECT* dont les colonnes récupérées seront les champs de fusion. La clause *WHERE* permet de filtrer sur l'enregistrement adéquat. Prenons par exemple cette requête :

```
SELECT employe.nom as nom_employe FROM employe WHERE employe.id = &idx
```

Elle est liée à la lettre-type *fiche_employe* comportant le champ de fusion *[nom_employe]*. Pour générer l'édition depuis l'objet métier *employe* on appelle :

```
$pdf = $this->compute_pdf_output('lettretype', 'fiche_employe', null, $this->getVal(
    ↪ $this->clePrimaire));
```

Note : Mettre à jour les colonnes de la requête SQL (*om_requete.requete*) oblige à corriger les champs de fusion (*om_requete.merge_fields*) afin de proposer une aide à la saisie cohérente lors de la rédaction d'une lettre-type.

4.8.2.4.1.2 OBJET

Dans le cas d'une requête objet les champs *om_requete.requete* et *om_requete.merge_fields* ne sont plus utilisés. Ce sont les méthodes de la classe *dbForm* *get_values_merge_fields()* et *get_labels_merge_fields()* qui retournent respectivement les champs de fusion et l'aide à la saisie. Pour reprendre l'exemple de la requête SQL, il faut dans ce cas spécifier la classe *employe* (*om_requete.classe*). Nous aurons automatiquement accès à tous ses champs. Les booléens sont transformés en oui/non, les dates sont formatées en jj/mm/aaaa et les libellés des clés étrangères sont récupérés.

Note : Si un employé est rattaché à une entreprise elle-même liée à une ville, alors définir la classe *employe;entreprise;ville* permettra de récupérer les champs de tous ces objets.

On peut également disposer de notre propre méthode (*om_requete.methode*) pour construire manuellement les champs de fusion. Celle-ci doit avoir un argument de type *string*. Il peut prendre la valeur de *values* ou *labels* selon où le framework y fait appel. Ainsi pour une requête objet dont on ne veut que le nom de l'employé il faut créer la méthode suivante :

```
public function get_only_name($type) {
    switch ($type) {
        case 'values':
            //
            $values = array();
            $values['employe.nom'] = $this->getVal('nom');
            return $values;
        case 'labels':
            //
            $labels = array();
            $labels['employe']['employe.nom'] = _("nom de l'employé");
            return $labels;
    }
}
```

Le tableau des labels a une dimension supplémentaire. Cela permet de catégoriser les champs de fusions proposés dans l'aide à la saisie (un tableau HTML par catégorie).

4.8.2.4.2 Modèle de données

```
CREATE TABLE om_requete
(
    om_requete integer NOT NULL, -- Identifiant unique
    code character varying(50) NOT NULL, -- Code de la requête
    libelle character varying(100) NOT NULL, -- Libellé de la requête
    description character varying(200), -- Description de la requête
    requete text, -- Requête SQL
    merge_fields text, -- Champs de fusion
    type character varying(200) NOT NULL, -- Requête SQL ou objet ?
    classe character varying(200), -- Nom de(s) la classe(s) contenant la méthode
    methode character varying(200), -- Méthode (de la première classe si plusieurs
    →définies) fournissant les champs de fusion. Si non spécifiée appel à une méthode
    →générique
    CONSTRAINT om_requete_pkey PRIMARY KEY (om_requete)
);
```

```
— obj/om_requete.class.php
— sql/pgsql/om_requete.form.inc.php
— sql/pgsql/om_requete.inc.php
— core/obj/om_requete.class.php
— core/sql/pgsql/om_requete.form.inc.php
— core/sql/pgsql/om_requete.inc.php
— gen/obj/om_requete.class.php
— gen/sql/pgsql/om_requete.form.inc.php
— gen/sql/pgsql/om_requete.inc.php
```

4.8.2.5 Les sous-états

Il est conseillé d'utiliser l'assistant sous-etat du générateur.

Les paramètres sont les suivants

```

texte et caractéristique du Titre
Intervalle avant et après le tableau
Entête de tableau (nom de colonne)
caractéristique du tableau
caractéristique des cellules
total, moyenne, nombre
requête SQL

```

Pour le titre, les zones entre crochets sont les champs de fusion, sélectionnés par la requête.

Les variables commençant par « & » sont celles définies dans dyn/varpdf.inc (exemple &aujourd'hui) et dans la table om_parametre.

- obj/om_sousetat.class.php
- core/obj/om_sousetat.class.php
- gen/obj/om_sousetat.class.php
- sql/pgsql/om_sousetat.form.inc.php
- sql/pgsql/om_sousetat.inc.php
- core/sql/pgsql/om_sousetat.form.inc.php
- core/sql/pgsql/om_sousetat.inc.php
- gen/sql/pgsql/om_sousetat.form.inc.php
- gen/sql/pgsql/om_sousetat.inc.php

4.8.2.6 Les champs de fusion

4.8.2.7 Les variables de remplacement

Lorsque dans les zones de remplacement des éditions, une chaîne de caractère commençant par « & » est identifiée elle essaye d'être remplacée. Ces éléments sont nommées variables de remplacement. Elles peuvent provenir de trois sources différentes :

- les fichiers de configuration ../dyn/var*pdf.inc,
- les méthodes globales de la classe du fichier ../obj/om_dbform.class.php,
- la table de paramètres om_parametre.

4.8.2.7.1 Les fichiers de configuration ../dyn/var*pdf.inc

4.8.2.7.2 Les méthodes globales de la classe du fichier ../obj/om_dbform.class.php

4.8.2.7.3 La table de paramètres om_parametre

prefixe_edition_substitution_vars doit être défini.

4.8.2.8 Les logos

- obj/om_logo.class.php
- core/obj/om_logo.class.php
- gen/obj/om_logo.class.php
- sql/pgsql/om_logo.form.inc.php
- sql/pgsql/om_logo.inc.php
- core/sql/pgsql/om_logo.form.inc.php
- core/sql/pgsql/om_logo.inc.php

- `gen/sql/pgsql/om_logo.form.inc.php`
- `gen/sql/pgsql/om_logo.inc.php`

4.8.2.9 L'éditeur WYSIWYG

Description de l'intégration de TinyMCE et des différentes configurations.

4.8.2.10 Les anciens fichiers de paramétrage

Les fichiers de paramétrage `../sql/pgsql/<OBJ>.etat.inc.php`, `../sql/pgsql/<OBJ>.etat.inc`, `../sql/pgsql/<OBJ>.lettretype.inc.php` ou `../sql/pgsql/<OBJ>.lettretype.inc` sont les anciens fichiers de paramétrage des éditions. Ils ne peuvent plus être utilisés depuis la version 4.0 du framework.

Un système d'import est disponible dans le générateur pour transformer ces anciens fichiers de paramétrage en enregistrement selon le nouveau format de paramétrage.

4.8.2.11 La prévisualisation

Le bouton Prévisualiser permet, pour une lettre type ou un état, d'avoir un aperçu du document qui sera généré. Les champs de fusion ne seront pas interprétés.

4.8.3 Les listings

4.8.3.1 Description de la fonctionnalité

4.8.3.2 Le fichier de paramétrage `../sql/pgsql/<OBJ>.pdf.inc.php`

Un état PDF peut être généré par le générateur (option).

Les paramètres sont les suivants

```
texte et caractéristique du Titre
Entête de tableau (nom de colonne)
caractéristique du tableau
caractéristique des cellules
total, moyenne, nombre
requête SQL
```

4.8.4 Les étiquettes

4.8.4.1 Description de la fonctionnalité

4.8.4.2 Le fichier de paramétrage `../sql/pgsql/<OBJ>.pdfetiquette.inc.php`

4.8.5 Composants

Les scripts du framework qui s'occupent de la gestion des éditions sont :

- `core/fpdf_etat.php`
- `core/fpdf_etiquette.php`
- `core/db_fpdf.php`
- `core/om_edition.class.php`

— scr/edition.php
 Les librairies PHP sont :
 — php/fpdf/
 — php/tcpdf/

4.9 Module “Import”

Sommaire

- *Module “Import”*
- *Principe*
- *Paramétrage*
 - *Le script de paramétrage ../sql/pgsql/<OBJ>.import.inc.php*
- *Composants*

4.9.1 Principe

Ce module permet l’import de données au format CSV dans les tables de la base de données par une interface utilisateur.

Exemple de format de fichier à importer (utilisateur.txt) :

```
nom,login,pwd,profil
"Georges DANDIN";"Georges";"21232f297a57a5a743894a0e4a801fc3";"3"
"Raymond DAVOS";"Raymond";"fe01ce2a7fbac8fafaed7c982a04e229";"3"
"Albert DUPONT";"Albert";"05c7e24700502a079cdd88012b5a76d3";"6"
```

4.9.2 Paramétrage

4.9.2.1 Le script de paramétrage ../sql/pgsql/<OBJ>.import.inc.php

Dans utilisateur.import.inc.php , il est defini :

```
le message affiché en import :
    $import= "Insert utilisateur" : Message

la table cible de l'import
    $table= "utilisateur"

la clé primaire si elle est automatique (mise en place d'une séquence) ;
ce champ est vide sinon
    $id="utilisateur"

Le verrouillage de la base de données
    $verrou = 1 mise-à-jour de la base
              = 0 pas de mise-à-jour pour une phase de test

Le mode debug
    $DEBUG =1 affichage des enregistrements à l'ecran
            =0 pas d'affichage
```

(suite sur la page suivante)

```

La mise en place d'un fichier d'erreur :
    $fic_erreur=1 fichier erreur
    =0 pas de fichier d'erreur

La mise en place d'un fichier de rejet reprenant les enregistrements csv rejetés
ce fichier contient les enregistrements en erreur et permet de relancer le
traitement après correction (manuelle)
    $fic_rejet=1 fichier de rejet pour relance traitement
    =0 pas de fichier rejet

La première ligne affiche le nom des champs :
$ligne1=1 la première ligne contient les noms de champs
    =0 sinon

Les zones obligatoires : tableau $obligatoire

    $obligatoire['nom']=1;// obligatoire = 1
    $obligatoire['login']=1;// obligatoire = 1

les tests d'existence d'une clé secondaire

    $exit['profil']=1 => 0=non / 1=oui
    $sql_exist["profil"] = "sélect profil from profil where profil = '"

La liste des champs à insérer
il faut mettre en commentaire les zones non traitées
    $zone['nom']='0' => la 1ère zone contient le nom
    $zone['login']=1 => la 2ème zone contient le login
    $zone['pwd']='2' => la 3ème zone contient le mot de passe (crypté)
    $zone['profil']='3' => la 4ème zone contient le profil

La valeur par défaut :
En effet, si $zone['profil']=" on peut définir un profil par défaut
    $default['profil']='5' Le profil par défaut sera 5

```

4.9.3 Composants

Les scripts du framework qui gèrent le module “Import” sont :

- core/om_import.class.php
- scr/import.php

4.10 Module “Reqmo”

Sommaire

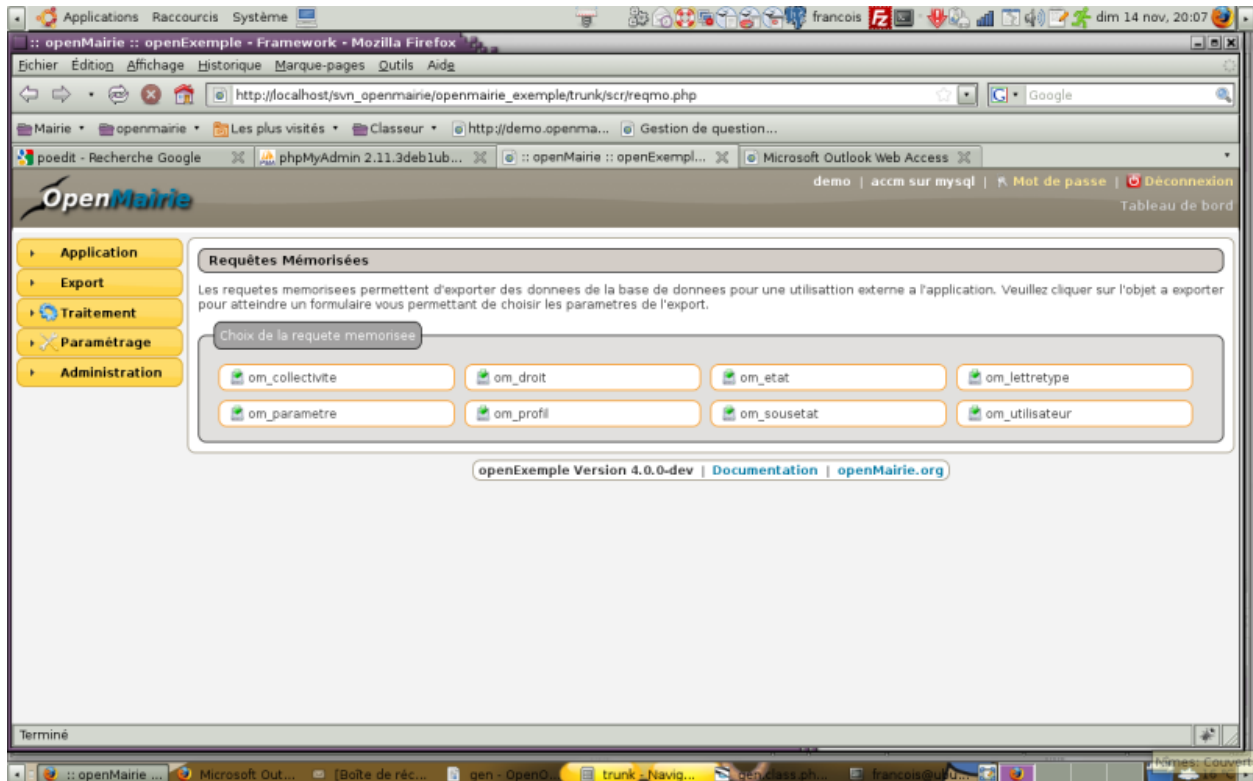
- *Module “Reqmo”*
 - *Principe*
 - *Paramétrage*
 - *Le script de paramétrage ../sql/pgsql/<OBJ>.reqmo.inc.php*
 - *Composants*

4.10.1 Principe

les requêtes mémorisées permettent au développeur de fournir un ensemble de requêtes :

- mémorisées
- accessible dans le menu export -> requêtes
- paramétrables par l'utilisateur
- permettant un affichage html en tableau ou un transfert au format csv sur tableur (choix du séparateur à l'utilisateur)

menu export-> requete



4.10.2 Paramétrage

4.10.2.1 Le script de paramétrage `../sql/pgsql/<OBJ>.reqmo.inc.php`

Les paramètres de reqmo sont :

`$reqmo["libelle"]` contient le libellé affiché en haut

`$reqmo["sql"]` contient la requête SQL.

Dans la requête, les paramètres sont mis entre []

et ils sont définis en dessous sous la forme `reqmo[parametre]=`.

« checked » : la colonne est affichée ou non

« un tableau » `array(a,b)` et le choix a ou b est donné à l'utilisateur de requête

« une requête sql » : le choix se fait dans la table du select

La requête exécutée est celle qui est reconstituée avec les zones saisies par l'utilisateur

voies sous openCimetiere

```
$reqmo['libelle']=" Voies par cimetiere ";

$reqmo['sql']=" select voie,voietype,voielib,
                [zonetype],[zonelib],
                [cimetierelib]
            from voie
            inner join zone
            on voie.zone=zone.zone
            inner join cimetiere
            on zone.cimetiere=cimetiere.cimetiere
            where cimetiere.cimetiere = [cimetiere] order by [tri]";

$reqmo['tri']= array('voielib',
                    'zonelib'
                    );
$reqmo['zonetype']="checked";
$reqmo['zonelib']="checked";
$reqmo['cimetierelib']="checked";
$reqmo['cimetiere']="select cimetiere,concat(cimetiere,' ',
                cimetierelib) from cimetiere";
```

4.10.3 Composants

Les scripts du framework qui gèrent le module “Reqmo” sont :

- core/om_reqmo.class.php
- scr/reqmo.php

4.11 Modules Géographiques

Nous présentons ici 2 modules géographiques

- le module d’affichage / saisie de données géographique : om_sig
- le module d’adresse automatique

4.11.1 Module om_sig

Ce module s’adresse aux développeurs et décrit le fonctionnement du sig interne

4.11.1.1 la nouvelle architecture

Une classe om_sig_map.class.php a été créé dans le core d’openMairie.

Attention la base de données om_sig_* a changée dans la version 4.4.5 (voir intégration 4.4.0 vers 4.4.5)

Les appels à la classe om_sig_map dans le core sont faits par :

- tab_sig.php
- form_sig.php
- les scripts spécifiques spg
- le script export_sig.php dans le repertoire scr
- le script reqmo ... pour appel a un tab_sig et unretour en format geojson

om_map_class appelle les fonctions javascripts contenus dans js/sig.js

4.11.1.1.1 tab_sig.php

les requêtes de mises à jour du geom ne sont plus initiée en form

les variables get sont les suivantes

```
obj : carte d'om_sig_map sur laquelle on travaille
idx : numéro d'enregistrement - peut être vide si on part d'une recherche
popup -> 0 : le menu est affiché; 1 : il ne l'est pas (tableau/options)
seli : géométrie sélectionnée
min max -> sert à la prépartion du geojson
etendue -> tableau options
reqmo -> tableau options
premier, recherche, selectioncol, adv_id -> tableau options
valide -> tableau options
style -> tableau options
onglet -> tableau options
adv_id=
recherche -> recherche simple exemple : 02
retour -> exemple : tab
```

appel librairie : « ../core/om_map.class.php »;

```
new om_map : obj + options
om_map = new om_map(obj, options);
om_map->recupOmSigMap();
om_map->recupOmSigflux();
om_map->computeFilters(options['idx']);
om_map->setParamsExternalBaseLayer();
om_map->prepareCanevas();
```

4.11.1.1.2 form_sig.php

les requêtes de mises à jour du geom ne sont plus initiée en form

variables get

```
obj, idx
popup -> tableau options
min max -> sert à la prépartion du geojson
etendue -> tableau options
reqmo -> tableau options
premier, recherche, selectioncol, adv_id -> tableau options
valide -> tableau options
style -> tableau options
onglet -> tableau options
validation (variable de validation spécifique form)
```

appel librairie : « ../core/om_map.class.php »;

```
New om_map : obj + options
om_map->recupOmSigMap
preparation du geojson
om_map->prepareForm( min, max, validation, geojson);
```

les fichiers scr/sig_json.php, sig_pannier.php, sig_wkt.php sont supprimés par rapport à la version 4.4.x

scr/travail_tab_sig.php???

4.11.1.1.3 spg/map_compute_geom.php

En utilisation du panier, computegeom fait une union des geometries lorsque l on fait une validation (bouton v) après selection de multiples géométries

variables get

```
obj, idx
popup -> tableau options
min max -> sert à la préparation du geojson
etendue -> tableau options
reqmo -> tableau options
premier, recherche, selectioncol, adv_id -> tableau options
valide -> tableau options
style -> tableau options
onglet -> tableau options
```

appel librairie : « ../core/om_map.class.php »;

```
om_map = new om_map(obj, options);
om_map->recupOmSigMap();
...
echo sep.om_map->getComputeGeom(c, geojson[i]);
```

4.11.1.1.4 spg/map_get_filters.php

gestion du filtre qgis dans om_sig_map_flux

Exemple le père et tous ses fils

```
SELECT 'fpere_point:²pere² IN ( '||pere||' );fpere_perim:²pere² IN ( '||pere||' );
↪ffils_point:²pere²
IN ( '||pere||' );ffils_perim:²pere² IN ( '||pere||' );ffils_perim:²pere² IN (
↪'||pere||' )'
AS buffer FROM ( SELECT array_to_string(array_agg(pere), ' , ') AS pere FROM &DB_
↪PREFIXEpere
WHERE pere IN (SELECT &idx::integer UNION &lst_idx) ) a
```

variables get

```
obj, idx
popup -> tableau options
min max -> sert à la préparation du geojson
etendue -> tableau options
reqmo -> tableau options
premier, recherche, selectioncol, adv_id -> tableau options
valide -> tableau options
style -> tableau options
onglet -> tableau options
```

appel librairie : « ../core/om_map.class.php »;

```
om_map = new om_map(obj, options);
om_map->recupOmSigMap();
om_map->recupOmSigflux();
...
om_map->computeFilters(idx_sel);
```

4.11.1.1.5 spg/map_get_geojson_cart.php

Récupère les géométries du panier

variables get

```
obj, idx
popup -> tableau options
min max -> sert à la préparation du geojson
etendue -> tableau options
reqmo -> tableau options
premier, recherche, selectioncol, adv_id -> tableau options
valide -> tableau options
style -> tableau options
onglet -> tableau options
```

appel librairie : « ../core/om_map.class.php »;

```
om_map = new om_map(obj, options);
om_map->recupOmSigMap();
om_map->recupOmSigflux();

lst=om_map->getGeoJsonCart(cart, lst);
... affichage de la liste
```

4.11.1.1.6 spg/map_get_geojson_markers.php

Renvoie en format json les marqueurs (ex bulles)

variables get

```
obj, idx
popup -> tableau options
min max -> sert à la préparation du geojson
etendue -> tableau options
reqmo -> tableau options
premier, recherche, selectioncol, adv_id -> tableau options
valide -> tableau options
style -> tableau options
onglet -> tableau options
```

appel librairie : « ../core/om_map.class.php »;

```
om_map = new om_map(obj, options);
om_map->recupOmSigMap();
lst=om_map->getGeoJsonMarkers(options['idx']);
... affichage de la liste
```

4.11.1.1.7 spg/map_redirection_onglet.php

Ce programme sert à faire afficher sous form : fenetre dans une fenetre courante

Il appelle utils.class.php

Il permet la redirection vers le formulaire de l'objet en visualisation (action=3) si l'objet existe et de faire un ajout sinon.

L'ajout ne fonctionne pas. En cas d'appui sur formulaire, le message apparait : « aucun enregistrement sélectionné »

4.11.1.1.8 scr/export_sig.php

ce programme permet de faire un affichage sur la base d'un tab ou d'une recherche.

- export suivant le moteur de recherche (équivalent de export csv)
- image dans app/img + app/css (appel à l'image)
- modification de core/om_layout.php fonction display_table_global_action
- pour l'instant ou csv ou sig -> pas de possibilité de faire les 2.
- bug en recherche si aucun enregistrement -> ouvre un nouvel onglet

4.11.1.1.9 scr/requeteur.php

La modification de programme permet de lancer un tab_sig depuis reqmo.

Le programme a donc évolué pour gérer cette possibilité mais elle n'est pas encore effective (voir alain)

Par contre une modification est obligatoire dans core/om_layout.class.php pour proposer l'option "json"

4.11.1.1.10 Nouvelles images dans img/ et nouvelle css pour l'interface om_sig

Des nouvelles images pour l'interface

```
map-nav.png
map-geoloc.png
map-form.png
map-edit-valid.png
map-edit-select.png
map-edit-record.png
map-edit-point.png
map-edit-modif.png
map-edit-get-cart.png
map-edit-erase.png
map-edit-draw-regular.png
map-edit-draw-polygon.png
map-edit-draw-line.png
map-edit.png
map-distance.png
map-area.png
```

des nouvelles images pour recherche csv ou sig (geojson)

```
sig.png
csv.png
```

Nouvelle css à mettre en layout_jqueryui_before.css

Problème d'affichage à régler

4.11.1.2 La classe om_map.class.php

Nouveauté version om 4.4.0sig

Ce module est dans CORE

Une classe intermédiaire est à créer dans obj/om_map.class.php

Actuellement créée, elle a un problème de prise en compte du zoom au départ BUG

4.11.1.2.1 Les variables globales

Elles sont définies ci dessous

```
var f;
    projections
    var defBaseProjection;
    var defDisplayProjection;

    paramètres
    var     obj;
    var     idx;
    var     sql_lst_idx;
    var     idx_sel;
    var     popup;
    var     seli;
    var etendue;
    var reqmo;
    var premier;
    var recherche;
    var selectioncol;
    var tricol;
    var advs_id;
    var valide;
    var style;
    var onglet;
    var type_utilisation = '';

    gestion de l'affichage
    var affichageZones= array();

    gestion de l'enregistrement
    var recordMultiComp; true: enregistrement de l'ensemble des champs géométriques ;
        false: enregistrement un par un des champs géométriques (par défaut)
    var recordMode; 1 (par défaut): via form_sig; 2 retour des valeurs dans des
        champs fournis dans le tableau recordFields
    var recordFields = array(); listes des champs retour (même index que comp)

    om_sig_map
    var sm_titre;
    var sm_source_flux;
    var sm_zoom;
    var sm_fond_sat;
    var sm_fond_osm;
    var sm_fond_bing;
    var sm_layer_info;
    var sm_fond_default;
    var sm_projection_externe;
    var sm_retour;
    var om_sig_map;
    var sm_url;
    var sm_om_sql;
    var sm_om_sql_idx;
```

(suite sur la page suivante)

(suite de la page précédente)

```
var sm_restrict_extent;
var sm_sld_marqueur;
var sm_sld_data;
var sm_point_centrage;

champs geom
var cg_obj_class = array();
var cg_maj = array();
var cg_table = array();
var cg_champ_idx = array();
var cg_champ = array();
var cg_geometrie = array();
var cg_lib_geometrie = array();

champs flux
var fl_om_sig_map_flux = array();
var fl_m_ol_map = array();
var fl_m_visibility = array();
var fl_m_panier = array();
var fl_m_pa_nom = array();
var fl_m_pa_layer = array();
var fl_m_pa_attribut = array();
var fl_m_pa_encaps = array();
var fl_m_pa_sql = array();
var fl_m_pa_type_geometrie = array();
var fl_m_sql_filter = array();
var fl_m_filter = array();
var fl_m_baselayer = array();
var fl_m_singletile = array();
var fl_m_maxzoomlevel = array();
var fl_w_libelle = array();
var fl_w_attribution = array();
var fl_w_id = array();
var fl_w_chemin = array();
var fl_w_couches = array();
var fl_w_cache_type = array();
var fl_w_cache_gfi_chemin = array();
var fl_w_cache_gfi_couches = array();

champs pour fonds de carte externes (OSM, Bing, Google)
var pebl_http_google;
var pebl_cle_bing;
var pebl_cle_google;
var pebl_zoom_osm_maj;
var pebl_zoom_osm;
var pebl_zoom_sat_maj;
var pebl_zoom_sa;
var pebl_zoom_bing_maj;
var pebl_zoom_bing;

paramètres de style pour la couche marqueur
var img_maj="../../img/punaise_sig.png";
var img_maj_hover="../../img/punaise_hover.png";
var img_consult="../../img/punaise_point.png";
var img_consult_hover="../../img/punaise_point_hover.png";
var img_w=14;
var img_h=32;
```

(suite sur la page suivante)

(suite de la page précédente)

```

var img_click="1.3";multiplicateur hauteur et largeur image cliquee

gestion des paniers
var cart_type = array(
    "point" => false,
    "line" => false,
    "polygon" => false
);
tableau de la barre du menu d'édition menu (id html, false)
var edit_toolbar= array(
    "#map-edit-nav" => false,
    "#map-edit-draw-point" => false,
    "#map-edit-draw-line" => false,
    "#map-edit-draw-polygon" => false,
    "#map-edit-draw-regular" => false,
    "#map-edit-draw-regular-nb" => false,
    "#map-edit-draw-modify" => false,
    "#map-edit-draw-select" => false,
    "#map-edit-draw-erase" => false,
    "#map-edit-cart" => false,
    "#map-edit-get-cart" => false,
    "#map-edit-draw-record" => false,
    "#map-edit-draw-delete" => false,
    "#map-edit-draw-close" => false
);

```

4.11.1.2.2 Les methodes

Les méthodes sont les suivantes

methode d'initialisation de l objet map :

```

construct(obj + options)
    initialisation des propriétés de l objet om_sig

    Récupération du paramétrage de l'objet dans les tables om_sig_map
et om_sig_map_comp. Préalable à toute utilisation de la classe
    function recupOmSigMap()
        requete om_sig_map + om_sig_extend

Génère un tableau (idx, sql_lst_idx) correspondant aux données idx/Reqmo/Recherche
le tableau est afficher par la classe om_table
    function getSelectRestrict(idx, seli)

    Génère un tableau GeoJson correspondant aux données idx/Reqmo/Recherche
    function getGeoJsonDatas(idx, seli)

    Génère un tableau GeoJson correspondant au panier cart (n de flux)
avec la liste des enregistrement lst
    function getGeoJsonCart(cart, lst)

    Génère un tableau GeoJson correspondant aux données idx/Reqmo/Recherche
    function getGeoJsonMarkers(idx)

calcul des filtres pour les flux de type WMS (fl_m_filter)

```

(suite sur la page suivante)

(suite de la page précédente)

```

function computeFilters(idx)

  Récupération du paramétrage des flux associés à l'objet dans les tables om_sig_
  ↪map_flux
  et om_sig_map_flux
function recupOmSigflux()
  requete dur om_sig_flux
  initialise le paramétrage de flux voir -> "champs flux"

  Initialisation des propriétés relatives aux fonds de carte externe,
  ajout des librairies associées si nécessaire
  function setParamsExternalBaseLayer()
  include de var_sig.inc
  initialise les "champs pour fonds de carte externes (OSM, Bing, Google)"

  Ecrit les propriétés de l'instance dans la page html pour JavaScript
  function prepareJS( )
  suite d'echo des variables

```

Preparation et affichage du canevas

```

Paramétrage des zones du canevas
function setCanevas(zone, val) {
    this->affichageZones[zone]=val;

  Préparation du canevas html: pilote les autres fonctions prepareCanevas...
  function prepareCanevas( )
  initialisation du tableau this->affichageZones

  Préparation du canevas html: menu avec regroupement (au moins une valeur à 1)
  function prepareCanevasMenu() {

Affichage du canevas MODIFIE JLB
  function prepareCanevasTitre()
  function prepareCanevasEdit()
  function prepareCanevasTools() -> fonction vidée par jlb
  function prepareCanevasInfos() -> fonction vidée par jlb
  function prepareCanevasPrint() -> fonction enlevée par jlb
  function prepareCanevasLayers() -> reprend les 2 fonctions enlevées par jlb
  function prepareCanevasNavigation() -> modif jlb
  function prepareCanevasGetfeatures() -> modif jlb

  Calcul la géométrie validé dans l'interface -> appel par spg/map_compute_geom.php
  function getComputeGeom(seli, geojson)

  Préparation du canevas html: pilote les autres fonctions prepareCanevas...
-> appel par form_sig.php
  function prepareForm( min, max, validation, geojson)p_init()

```

voir core/obj et core/sql

4.11.1.3 Le java script

les fonctions sont dans js/sig.js

Nouvelles fonctions

```

affectation d'un mode d'action :
    Géolocalisation, Information,
    Navigation, Edition, Formulaire,
    Mesure surface, Mesure distance
    function map_set_mode_action(mode)

function popupIt pour utilisation dans sig
    function map_popupIt(objsf, link, width, height, callback, callbackParams) {

implémentation simplifiée de l'utilisation de popupIt dans la fonction SIG
function map_popup(url

vide les div du div map-getfeatures
function map_clear_getfeatures()

ouverture du div map-getfeatures
function map_close_getfeatures()

affichage dans la zone info
function map_affiche_info(message)

affichage dans la zone titre
function map_affiche_titre()

Traitement du select d'un élément de la couche des marqueurs
function map_on_select_feature_marker(event)

Traitement du unselect d'un élément de la couche des marqueurs
function map_on_unselect_feature_marker(event)

Traitement du select d'un élément de la couche des datas
function map_on_select_feature_data(event)

Traitement du unselect d'un élément de la couche des datas
function map_on_unselect_feature_data(event)

récupère la valeur d'un attribut (traitement GetFeatureInfo)
function traiteGetFeatureInfoRecAttribut(attributes,name)

récupère la valeur d'un attribut et formate la restitution (traitement GetFeatureInfo)
function traiteGetFeatureInfoRecAttributFormat(attributes,name,title, formatage)

traitement pour chaque GetFeatureInfo
function traiteGetFeatureInfo(i,rText,res)

Traitement du clic sur la carte
function map_clic(e)

copie les données geoson correspondant à l'idx sélectionné dans les
couches d'édition correspondantes
function map_copyDataToEditLayers()

Traitement clic Edit Draw Point
function map_clicEditDrawPoint()

Traitement clic Edit Draw Line
function map_clicEditDrawLine()

```

(suite sur la page suivante)

(suite de la page précédente)

```
Traitement clic Edit Draw Polygon
function map_clcEditDrawPolygon()

//traitement du onChange sur le nombre de côté du polygone régulier
function map_EditDrawRegularChange()

Traitement clic Edit Draw Polygon
function map_clcEditDrawRegular()

Traitement clic Edit modify
function map_clcEditDrawModify()

Traitement clic Edit select
function map_clcEditSelect()

Traitement clic Edit navigate
function map_clcEditNavigate()

Traitement clic Edit Erase
function map_clcEditErase()

test si une valeur est un entier
function map_is_int(value)

désactive tous les controles d'édition sauf sauf
function map_deactivate_all_edit_control(sauf)

affiche/cache les éléments de la barre d'outils d'édition
function map_RefreshEditBar()

  calcule et valide les champs géométriques dont les composants sont sélectionnés
function map_computeGeom(sel, enreg)

traitement du clic sur le bouton de récupération du panier
function map_clcEditGetCart()

Selection d'un panier
function map_select_cart(n)

Selection d'un champ d'édition
function map_select_edit_champ(n)

Traitement géolocalisation
function map_clcGeolocate()

traitement bouton Edit
function map_clcEdit()

traitement bouton Edit Close
function map_clcEditClose()

Traitement de l'application des SLD pour les données geojson
function map_successGetSld_geojson_datas(req)

Chargement des données aux formats geojson
function map_load_geojson_datas(bRefreshFilters)
```

(suite sur la page suivante)

(suite de la page précédente)

```

Traitement de l'application des SLD pour les marqueurs geojson
function map_successGetSld_geojson_markers(req)

Chargement des marqueurs aux formats geojson
function map_load_geojson_markers()

fonction d'identification des tuiles pour les flux de type SMT
function map_flux_SMT(bounds)

Initialisation des flux (om_sig_map_flux)
function map_load_flux(i)

Chargement des flux de type base
function map_load_bases_layers()
    fonction qui charge les fonds osm, bing, sat ou un numéro de flux

vide une des listes du selecteur (div map-layers)
(dest: baselayers, overlays, datas, markers)
function map_empty_layer_list(dest)

Chargement des flux de type overlays
function map_load_overlays()

Chargement des flux de type cart
function map_load_carts()

Affichage de la couche de base sélectionnée
function map_display_base_layer()

Ajoute du control openlayers de géolocalisation
function map_addGeolocateControl()

Ajoute le controle openLayers de sélection
function map_add_SelectControl()

Traitement des controles de mesure
function map_handleMeasurements(event)

Ajoute des controles de mesure
function map_add_MeasureControls()

Ajoute les contrôles openLayers à la carte
function map_add_controls()

Initialisation de la carte
function map_init()

```

affichage en onglet (jlb)

```

affiche_aide
affiche_layers
affiche_tools
affiche_baselayers
affiche_getfeatures

```

A voir

```
pb d affichage info si utilisation boite a outil : navigation ou geoloc ou
mesure distance ou mesure aire.
```

4.11.1.4 Integration de la version 4.4.x en 4.4.5 :

Les principes de la verion om4.5.x sont les suivants :

- cohérence de la liaison de un à plusieurs : om_sig / maj de champs
- utilisation d'om_sig dans le moteur de recherche et reqmo
- création d'une table d'extension remplaçant var_sig.inc
- la table de flux remplace celle des wms car il y a plusieurs type de flux

Pour faire évoluer la base, utiliser data/pgsql/ver_4.4.5.sql

Il est décrit ici les modifications de la base :

Nouveaux champs om_sig_map

```
champ_idx character varying(30);
util_idx boolean;
util_reqmo boolean;
util_recherche boolean;
source_flux integer;
fond_default character varying(10); Paramétrage du champ par défaut

sld_marqueur character varying(254);
sld_data character varying(254);
point_centrage geometry(Point,2154),;
```

Dans om_sig_map_comp, sont intégrés la partie mise à jour d om_sig_map (respect du modèle référentiel)

Ces colonnes relatives à la mise à jour sont supprimées dans om_sig_map :

```
lib_geometrie;
maj;
table_update;
champ_idx;
champ;
type_geometrie;
```

Création d'une table des extensions (remplaçant les variables de dyn/var_sig.inc.php) et clé secondaire dans om_sig_map :

```
om_sig_extent integer NOT NULL,
nom character varying(150),
extent character varying(150)
```

Creation d'une table om_sig_flux remplaçant om_sig_wms car les flux peuvent être différents que wms

```
om_sig_flux integer NOT NULL,
libelle character varying(50) NOT NULL,
om_collectivite integer NOT NULL,
id character varying(50) NOT NULL,
attribution character varying(150),
chemin character varying(255) NOT NULL,
couches character varying(255) NOT NULL,
cache_type character varying(3),
cache_gfi_chemin character varying(255),
cache_gfi_couches character varying(255)
```

Dans le modèle, remplacement de tous les champs oui/non par des types booléens

4.11.1.5 Reprise des évolutions de la 4.4.0

Il s'agit des évolutions faites sur opencadastre et openAdresse sur la base de la 4.4.3 à intégrer dans la version om_sig 4.4.5

4.11.1.5.1 openCadastre

- utilisation de dyn/tab_sig_barre.inc -> boîte appelé par une flèche
- modification du app/js/sig.js -> envoi en ajax de recherche avec localisation
- en app des fichiers utilisant fpdf
- Taille de fenêtre « carto » identique à fenêtre d'appel
- Taille de la carte s'adapte à la fenêtre affichée
- persistance du zoom + fond + overlay -> utilisation des variables sessions
- correction de la projection temporaire 900913 -> compatibilité qgis
- flag dans var.sig pour utiliser l'ancienne projection

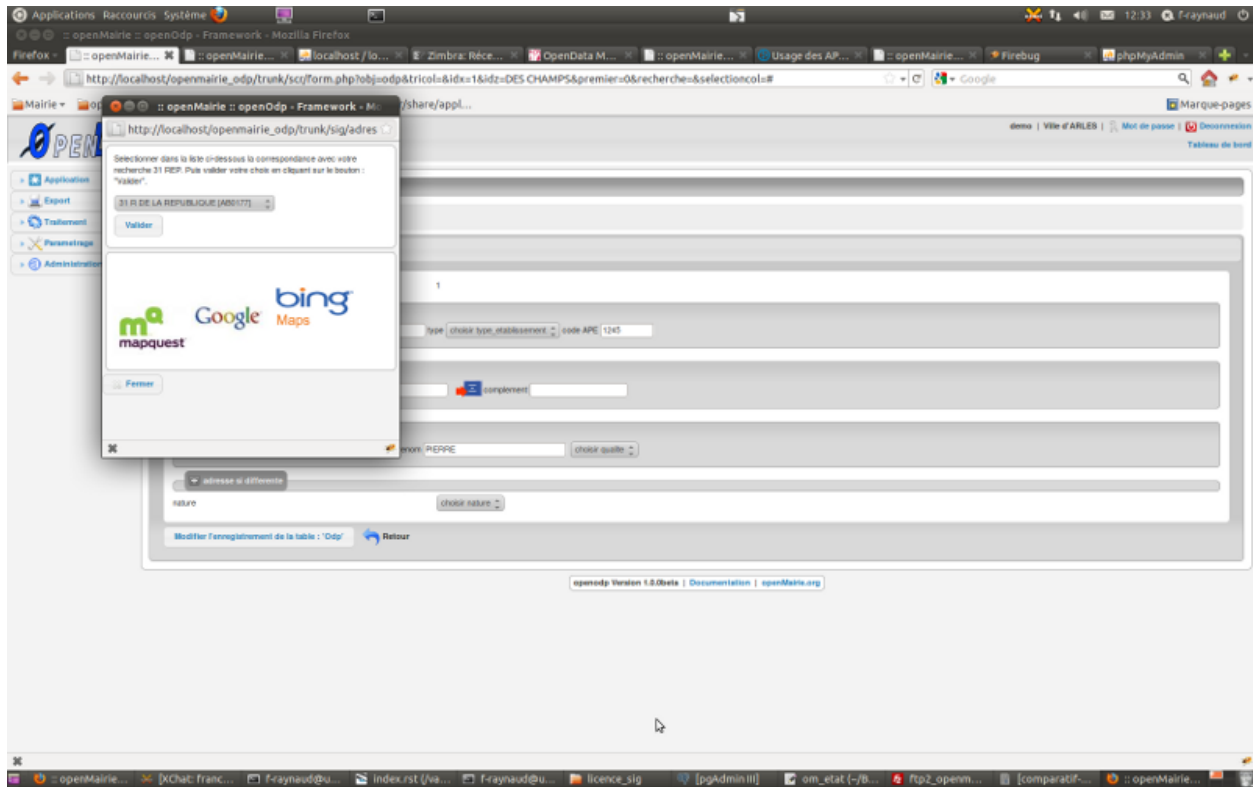
4.11.1.5.2 openAdresse

A voir

4.11.2 Module d'adresse

Il s'agit du module permettant de récupérer une adresse

Il est basé sur le script



Il est appelé depuis la classe métier suivant l'exemple suivant :

Exemple de openmairie_domainepublic : objet odp

dans sql/pgsql/odp.form.inc : le champ adressepostale est implementé comme un champ_

→ vide

```
$champs=array("odp", ...
              "' as adresse_postale", // specific
```

dans obj/odp.class.php

dans la methode setType, le champ adresse_postale est du type httpclick

```
function setType (&$form, $maj) {
    parent::setType ($form, $maj);
    $form->setType('adresse_postale', 'httpclick');
```

avec la methode setVal : valoriser par défaut l'accès au script adresse_postale
app/js/script.js

```
function setVal(&$form, $maj, $validation, &$db, $DEBUG=null){
    // bouton adresse postale
    $form->setVal("adresse_postale",
        "adresse_postale('f1',f1.libelle_voie.value,f1.numero_voie.value)");
}
```

Initialiser une variable globale égale à 0 et qui prend la valeur 1 si la zone_

→ geometrique

est au format wkt

En effet le point ramené par l'API externe est au format géographique (latitude, _

→ longitude) en wkt

(suite sur la page suivante)

(suite de la page précédente)

il commence par POINT(x, y) et il convient de le mettre dans la projection de la zone_
 ↳ géométrique de la table ODP

```
class odp extends odp_gen {
```

```
    var $wkt=0;
```

dans la methode setValF, repérer une valeur wkt

```
    if(substr($val['geom'],0,5)== "POINT"){
        $this->wkt=1;
        $this->valF['geom'] = null;
    } ...
```

utiliser les methodes de mise à jour après saisie pour la geometrie :

```
function triggermodifierapres($id,&$db,$val,$DEBUG) {
    if($this->wkt==1){
        $this->sig_wkt($id,&$db,$val,$DEBUG);
    }
}
```

```
function triggerajouterapres($id,&$db,$val,$DEBUG) {
    $id=$this->valF[odp]; // id n est pas valorise en ajout
    if($this->wkt==1){
        $this->sig_wkt($id,&$db,$val,$DEBUG);
    }
}
```

```
function sig_wkt($id,&$db,$val,$DEBUG){
    // si wkt -> saisie en format binaire wkb pour postgre
    $projection = $db -> getOne("select srid from geometry_columns where f_table_
↳ name='".
    $this->table."'");
    $sql ="update ".$this->table." set geom =geometryfromtext('".$val["geom"]."',
↳ ".
    $projection." ) where ".$this->table." ='".$id."'";
    $res = $db -> query($sql);
    if (DB :: isError($res)){
        die($res->getMessage()."erreur ".$sql);
    }else{
        $this->msg = $this->msg."&nbsp;".$_("le point trouvé par l'API est_
↳ sauvegardé")."&nbsp;";
        $this->table."&nbsp;".$_.$id;
    }
}
```

4.12 Abstraction du “layout” (ergonomie)

Avertissement : Cette rubrique est en cours de rédaction.

Sommaire

- *Abstraction du “layout” (ergonomie)*
 - *Le composant jquery*
 - *Les feuilles de style*

Depuis la version openMairie 4, il est utilisé l’ergonomie de jquery.

4.12.1 Le composant jquery

Les skins jquery peuvent être rajoutés dans le répertoire /om_theme/.

Il est possible de générer un nouveau thème directement depuis le site de JQuery-UI

```
http://jqueryui.com/download    pour télécharger un thème standard
http://jqueryui.com/themeroller pour créer un thème personnalisé
```

Le changement de thème peut se faire dans le fichier EXTERNALS.txt

voir framework/paramétrage

4.12.2 Les feuilles de style

Les feuilles de style sont stockées dans le repertoire css/ et sont cascadables

```
main.css      : feuille de style principale openMairie
om-theme/om.css : suivant la feuille de style jquery (voir EXTERNALS.txt)
app/css       : surcharge spécifique a l'application (exemple : le logo de
                l'application)
```

4.13 Abstraction du “filestorage” (stockage des fichiers)

Sommaire

- *Abstraction du “filestorage” (stockage des fichiers)*
 - *Principe général*
 - *Fonctionnement*
 - *Description de l’abstracteur*
 - *Description du fichier de configuration*
 - *Description des méthodes de la classe filestorage*
 - *Description du connecteur **depredacted***
 - *Description du connecteur **filesystem***
 - *Description du connecteur **filetransferprotocol***
 - *Description du connecteur **alfresco***
- *Utilisation*
 - *Configuration du widget Upload*
 - *Contraintes*
 - *Métadonnées*
 - *Récupération du fichier*
 - *Scripts permettant de visualiser / d’accéder au fichier*

— *spg/file.php*
 — *spg/voir.php*

4.13.1 Principe général

L'objectif de cette rubrique est de gérer le stockage des fichiers dans les applications. Ce stockage permet d'avoir des stockages complexes.

Termes :

- **abstracteur** : classe d'abstraction, c'est elle qui est instanciée et qui instancie les classes spécialisées en fonction des critères,
- **connecteur** : classe spécialisée.

Description des fichiers permettant de gérer le filestorage :

```
[D] dyn
|-[F] filestorage.inc.php
|---- ...
[D] core
|-[F] om_filestorage.class.php
|-[F] om_filestorage_deprecated.class.php
|-[F] om_filestorage_filesystem.class.php
|-[F] om_filestorage_filetransferprotocol.class.php *
|-[F] om_filestorage_alfresco.class.php *
|---- ...

* (ce fichier n'existe pas il est là à titre d'illustration de ce que le système est
↳ capable de faire)
```

4.13.2 Fonctionnement

4.13.2.1 Description de l'abstracteur

[core/om_filestorage.class.php]

Ce fichier est composé de deux classes : la classe d'abstraction, la classe de base pour les classes spécialisées.

La classe “filestorage” est une classe d'abstraction de stockage de fichiers. C'est cette classe qui est instanciée et utilisée par d'autres scripts pour gérer la création, récupération, suppression de fichiers et ce peu importe le stockage utilisé. Son objectif est d'instancier la classe de stockage spécifique aussi appelée plugin de stockage correspondant au paramétrage sélectionné. Cette classe de stockage spécifique hérite de la classe “filestorage_base” qui lui sert de modèle.

4.13.2.2 Description du fichier de configuration

[dyn/filestorage.inc.php]

Ce fichier de configuration doit permettre le paramétrage du stockage dans chacune des applications. Par exemple, il permet dans une installation d'openElec de stocker les fichiers en utilisant le connecteur “filesystem” dans le path /var/openelec/data sur le système de fichiers et dans une autre installation d'openElec de stocker les fichiers dans “alfresco” sur un autre serveur.

Il doit permettre de sélectionner le stockage à utiliser ainsi que le paramétrage spécifique à chacun des connecteurs.

Par exemple : pour le stockage “filesystem”, il doit être possible de paramétrer le répertoire dans lequel vont être stockés les fichiers, pour le stockage “alfresco” il doit être possible de paramétrer l’adresse de l’hôte, l’identifiant et le mot de passe de connexion, ...

Il permet aussi de stocker la configuration du stockage de fichier temporaire.

Ce fichier s’inspire des autres fichiers de configuration (mail.inc.php, directory.inc.php, ...). Il doit contenir un tableau associatif :

```
$filestorage = array();
```

Exemple d’un paramétrage (la clé filestorage-default doit être rajoutée dans une des valeurs du paramétrage du fichier database.inc.php comme l’est mail-default ou directory-default) :

```
$filestorage["filestorage-default"] = array {
    "storage" => "filesystem", // l'attribut storage est obligatoire
    "storage_path" => "", // l'attribut storage_path n'est pas obligatoire
    "path" => "/var/www/openfoncier/data/",
    "temporary" => array(
        "storage" => "filesystem", // l'attribut storage est obligatoire
        "path" => "../tmp/", // le repertoire de stockage
    )
    ...
}
```

Si aucun filestorage n’est paramétré, le filestorage deprecated sera instancié et le filestorage temporaire sera le filesystem.

4.13.2.3 Description des méthodes de la classe filestorage

La classe “filestorage” contient des méthodes de gestion des fichiers :

filestorage.create (\$data, \$metadonnees, \$mode = "from_content")

Permet de créer un fichier sur le filestorage,

\$data : contenu du fichier

\$metadonnees : tableau contenant la liste des métadonnées (\$cle => valeur)

\$mode [« from_content », « from_path »] :

— from_content : \$data contient le contenu du fichier.

— from_temporary : \$data l’uid d’un fichier enregistré sur le filesystem temporary.

— from_path : \$data contient le chemin du fichier à enregistrer.

Cette méthode retourne l’UUID du fichier enregistré.

filestorage.update (\$uid, \$data, \$metadonnees, \$mode = "from_content")

Permet de mettre à jour un fichier sur le filestorage,

\$data : contenu du fichier

\$metadonnees : tableau contenant la liste des métadonnées (\$cle => valeur)

\$mode [« from_content », « from_path »] :

— from_content : \$data contient le contenu du fichier.

— from_temporary : \$data l’uid d’un fichier enregistré sur le filesystem temporary.

— from_path : \$data contient le chemin du fichier à enregistrer.

Cette méthode retourne l’UUID du fichier enregistré.

filestorage.get (\$uid)

Cette méthode retourne le contenu et les métadonnées d’un fichier en fonction de l’UUID passé en paramètre.

filestorage.delete (\$uid)

Cette méthode supprime un fichier en fonction de l’UUID passé en paramètre.

`filestorage.create_temporary($data, $metadonnees, $mode = "from_content")`

Permet de créer un fichier sur le filestorage temporaire,

\$data : contenu du fichier

\$metadonnees : tableau contenant la liste des métadonnées (\$cle => valeur)

\$mode [« from_content », « from_path »] :

— from_content : utilisation normale de la méthode create(), \$data contient le contenu du fichier.

— from_path : \$data contient le chemin du fichier à enregistrer.

Cette méthode retourne l'UUID du fichier enregistré temporairement.

`filestorage.get_temporary($uid)`

Cette méthode retourne le contenu et les métadonnées d'un fichier enregistré temporairement en fonction de l'UUID passé en paramètre.

`filestorage.delete_temporary($uid)`

Cette méthode supprime un fichier temporaire en fonction de l'UUID passé en paramètre.

L'appel aux méthodes « temporary » se fait sur une instance de filesystem défini dans le paramétrage. Ces méthodes sont implémentés dans la classe de base contrairement aux autres méthodes, elle peuvent toutefois être surchargées dans les classes de connecteurs spécifiques.

4.13.2.4 Description du connecteur deprecated

[core/om_filestorage_deprecated.class.php]

Cette classe est une classe de stockage spécifique aussi appelée plugin de stockage pour le système d'abstraction de stockage des fichiers. Le principe de ce plugin est de stocker tous les fichiers à plat selon la méthode utilisée avant la création du système de stockage. Ce plugin a été créé uniquement dans un soucis de garder la compatibilité pour les applications existantes.

4.13.2.5 Description du connecteur filesystem

[core/om_filestorage_filesystem.class.php]

Cette classe est une classe de stockage spécifique aussi appelée plugin de stockage pour le système d'abstraction de stockage des fichiers. Le principe de ce plugin est de stocker tous les fichiers en renommant le fichier avec un UUID (identifiant unique) et en créant une arborescence à deux niveaux. Le premier est composé des deux premiers caractères de l'UUID du fichier et le second niveau des quatre premiers caractères de l'UUID du fichier. Un fichier avec l'extension .info permet de stocker les informations de base du fichier ainsi que des métadonnées.

Schéma du stockage :

```
/repertoire/de/stockage/25/252e/252ece72d4c0f88782d9fd6b99f43dfd

/repertoire/de/stockage/ :
/25/ : Le premier niveau des dossiers contenant les deux premiers caractères de l'
↳ uuid du fichier, la méthode de génération des uuid est fourni dans la suite du
↳ paragraphe
/252e/ : Le second niveau des dossiers contenant les 4 premiers caractères de l'uuid
↳ du fichier, la méthode de génération des uuid est fourni dans la suite du paragraphe
252ece72d4c0f88782d9fd6b99f43dfd : Le fichier est stocké avec pour nom un uuid sans
↳ extension, la méthode de génération des uuid est fourni dans la suite du paragraphe

252ece72d4c0f88782d9fd6b99f43dfd.info : Les fichiers .info sont là pour stocker les
↳ métadonnées de chaque fichier, ce sont des fichiers textes qui sont formatés :

# trois informations obligatoires (ces commentaires ne doivent pas apparaître dans le
↳ fichier .info)
```

(suite sur la page suivante)

(suite de la page précédente)

```
filename="plop.pdf"
mimetype="application/pdf"
size="124541"

# métadonnées supplémentaires facultatives (ces commentaires ne doivent pas
↳ apparaître dans le fichier .info)

propriete1="valeur1"
propriete2="valeur2"

252ece72d4c0f88782d9fd6b99f43dfd_lock : Les fichiers _lock sont là pour servir de
↳ marqueur et indiquer si le fichier est locké ou non.
```

Exemple d'arborescence de stockage :

```
/repertoire/de/stockage/25/252e/252ece72d4c0f88782d9fd6b99f43dfd
/repertoire/de/stockage/25/252e/252ece72d4c0f88782d9fd6b99f43dfd.info
/repertoire/de/stockage/25/252e/252ece72d4c0f88782d9fd6b99f43dfd_lock
/repertoire/de/stockage/25/252e/252eacd35ef547dab12ded6b99f43dfd
/repertoire/de/stockage/25/252e/252eacd35ef547dab12ded6b99f43dfd.info
/repertoire/de/stockage/25/252e/252eacd35ef547dab12ded6b99f43dfd_lock
/repertoire/de/stockage/12/123a/123aacd35ef547dab12ded6b99f43dfd
/repertoire/de/stockage/12/123a/123aacd35ef547dab12ded6b99f43dfd.info
/repertoire/de/stockage/12/123a/123aacd35ef547dab12ded6b99f43dfd_lock
```

Méthode pour générer les uuid :

```
function generate_uuid($prefix = "") {
    return md5(uniqid($prefix, true));
}
```

4.13.2.6 Description du connecteur filetransferprotocol

[core/om_filestorage_filetransferprotocol.class.php]

Ce fichier permet de déclarer la classe spécialisée pour stocker les fichiers sur un FTP (ce fichier n'existe pas il est là à titre d'illustration de ce que le système est capable de faire).

4.13.2.7 Description du connecteur alfresco

[core/om_filestorage_alfresco.class.php]

Ce fichier permet de déclarer la classe spécialisée pour stocker les fichiers sur Alfresco (ce fichier n'existe pas il est là à titre d'illustration de ce que le système est capable de faire).

4.13.3 Utilisation

Les méthodes de la classe d'abstraction sont désormais utilisées dans la classe upload et dans les widgets upload file du formulaire.

Il est possible de paramétrer une liste de métadonnées d'un champ upload, certains champs de ce formulaire pouvant contenir certaines informations à ajouter aux informations du fichier uploadé, il est nécessaire de créer le fichier lors

de la validation du formulaire. Pour ce faire le fichier uploadé sera enregistré temporairement sur le filestorage défini pour les fichiers temporaires puis enregistré sur le filestorage définitif lors de la validation du formulaire.

Hors formulaire la méthode create peut être utilisée de 3 façons :

- en lui passant le chemin du fichier dans \$data et avec le mode défini à « from_path »
- en lui passant le contenu du fichier dans \$data (fonctionnement existant avant modification)
- en lui passant l'UUID d'un fichier temporaire avec le mode défini à « from_temporary »

4.13.3.1 Configuration du widget Upload

4.13.3.1.1 Contraintes

Les contraintes sont à rajouter dans la classe métier de l'objet concerné, dans la méthode setSelect.

Exemple de configuration de l'ajout de contraintes de contrôles de la taille maximale et de l'extension lors de l'upload de fichier :

```
<?php
    $params = array(
        "constraint" => array(
            "size_max" => 2,
            "extension" => ".pdf;.txt;.odt"
        ),
    );
?>
```

La taille maximale est en mo et la liste des extensions est une chaîne de caractères.

4.13.3.1.2 Métadonnées

Il y a des métadonnées globales et spécifiques.

Les globales sont définies dans [obj/om_db_form.class.php] dans l'attribut \$metadata_global.

Exemple de configuration :

```
<?php
    var $metadata_global = array(
        "metadonne1" => "methodeQuiRetourneLaBonneValeur1",
        "metadonne2" => "methodeQuiRetourneLaBonneValeur2",
    );
?>
```

Les spécifiques sont à ajouter en attribut de la classe métier de l'objet concerné.

Exemple de configuration de l'ajout de métadonnées :

```
<?php
    var $metadata = array(
        "champ" => array(
            "metadonne1" => "methodeQuiRetourneLaBonneValeur1",
            "metadonne2" => "methodeQuiRetourneLaBonneValeur2",
        ),
    );
?>
```

Les clés de ces tableaux sont les noms des métadonnées, les valeurs associées sont les noms des méthodes qui retournent les métadonnées.

4.13.3.2 Récupération du fichier

```
//  
$file = $f->storage->get($fic);
```

4.13.3.3 Scripts permettant de visualiser / d'accéder au fichier

4.13.3.3.1 spg/file.php

Le script permet de télécharger le fichier.

Le code pour composer le lien vers ce script est le suivant :

```
<?php  
  
$file_download_link = "../spg/file.php?";  
if ($obj != "" && $champ != "" && $id != "") {  
    $file_download_link .= "obj=".$obj."&champ=".$champ."&id=".$id;  
} else {  
    $file_download_link .= "uid=".$fic;  
}  
  
?>
```

4.13.3.3.2 spg/voir.php

Le script permet de visualiser le fichier.

```
<?php  
  
$file_voir_link = "../spg/voir.php?";  
if ($obj != "" && $champ != "" && $id != "") {  
    $file_voir_link .= "obj=".$obj."&champ=".$champ."&id=".$id;  
} else {  
    $file_voir_link .= "uid=".$fic;  
}  
  
?>
```

5.1 Pré-requis

5.1.1 PHPUnit

<https://phpunit.de>

5.1.1.1 Installation

```
wget https://phar.phpunit.de/phpunit.phar
chmod +x phpunit.phar
sudo mv phpunit.phar /usr/local/bin/phpunit
```

5.1.2 Selenium Server

<http://www.seleniumhq.org>

5.1.2.1 Installation

```
wget http://selenium-release.storage.googleapis.com/2.53/selenium-server-standalone-2.
↪53.1.jar
java -jar selenium-server-standalone-2.53.1.jar
```

5.1.3 Robot Framework

<http://robotframework.org/>

5.1.3.1 Installation

```
sudo apt-get install python python-dev python-pip libjpeg-dev
sudo pip install selenium robotframework robotframework-selenium2library_
↳ robotframework-selenium2screenshots requests robotframework-requests Pillow_
↳ robotframework-archivelibrary
```

Avertissement : Il faut utiliser la version 2.7 de python.

Avertissement : La librairie selenium2screenshots doit être modifiée pour fonctionner correctement.

```
vim /usr/local/lib/python2.7/dist-packages/Selenium2Screenshots/keywords.robot
```

```
:%s/u' '/u' ' count=1/g
```

5.1.4 Jenkins

jenkins.openmairie.org

5.2 Arborecence du répertoire *tests*

```
.
├── [drwxrwxr-x] binary_files
├── [drwxrwxr-x] doc
├── [drwxrwxr-x] results
├── [drwxrwxr-x] resources
│   ├── [drwxrwxr-x] app
│   │   ├── [drwxrwxr-x] gen
│   │   ├── [-rw-rw-r--] __init__.py
│   │   ├── [-rw-rw-r--] om_tests.py
│   │   └── [-rw-rw-r--] keywords.robot
│   ├── [drwxrwxr-x] core
│   │   ├── [-rw-rw-r--] __init__.py
│   │   └── [-rw-rw-r--] resources.robot
├── [-rwxrwxr-x] om-tests
├── [-rw-rw-r--] config.xml
├── [-rw-rw-r--] 000_generation.robot
└── [-rw-rw-r--] 000_test_unitaire.php
```

5.2.1 *tests/om-tests*

Ce fichier doit être exécutable.

```
#!/usr/bin/python
from resources.app.om_tests import om_tests
tests = om_tests()
tests.main()
```

5.2.2 tests/config.xml

```
<phpunit>
  <testsuites>
    <testsuite name="openmairie">
      <file>000_test_unitaire.php</file>
    </testsuite>
  </testsuites>
</phpunit>
```

5.2.3 tests/000_generation.robot

```
*** Settings ***
Resource    resources/resources.robot
Suite Setup  For Suite Setup
Suite Teardown  For Suite Teardown
Documentation  Le 'Framework' de l'application permet de générer
...   automatiquement certains scripts en fonction du modèle de données. Lors
...   du développement la règle est la suivante : toute modification du
...   modèle de données doit entraîner une régénération complète de tous les
...   scripts. Pour vérifier à chaque modification du code que la règle a bien
...   été respectée, ce 'Test Suite' permet de lancer une génération complète.
...   Si un fichier est généré alors le test doit échoué.

*** Test Cases ***
Génération complète

    Depuis la page d'accueil    admin    admin
    Générer tout
```

5.2.4 tests/000_test_unitaire.php

```
<?php
class General extends PHPUnit_Framework_TestCase {

    /**
     * Méthode lancée en début de traitement
     */
    public function setUp() {
    }

    /**
     * Méthode lancée en fin de traitement
     */
    public function tearDown() {
    }

    /**
     * Test Case n°01
     */
    public function test_case_01() {
        require_once "../obj/utils.class.php";
```

(suite sur la page suivante)

(suite de la page précédente)

```

    @session_start();
    $_SESSION['collectivite'] = 1;
    $_SESSION['login'] = "admin";
    $_SERVER['REQUEST_URI'] = "";
    $f = new utils("nohtml");
    $f->disableLog();

    $this->assertEquals($year, 2015);

    $f->__destruct();
}
?>

```

5.2.5 tests/doc/

Répertoire destiné à recevoir la génération de la documentation des mots clés Robot Framework.

5.2.6 tests/results/

Répertoire destiné à recevoir la génération des rapports et des captures d'écran produits pendant l'exécution des tests. Afin que ces nouveaux fichiers ne gênent pas l'utilisation des commandes Subversion, tous les fichiers à l'intérieur de ce dossier sont ignorés grâce à la propriété svn :ignore.

5.2.7 tests/binary_files/

Répertoire destiné à recevoir les fichiers de configuration ou d'initialisation de l'environnement de tests.

5.2.8 tests/resources/

Répertoire contenant les ressources utilisées par les tests suite.

5.2.9 tests/resources/__init__.py

Fichier vide pour définir le répertoire *resources* comme un module python.

5.2.10 tests/resources/resources.robot

```

*** Settings ***
#
Resource      core${/}om_resources.robot
#
Resource      app${/}keywords.robot

*** Variables ***
${SERVER}      localhost
${PROJECT_NAME}  openexemple
${BROWSER}     firefox

```

(suite sur la page suivante)

(suite de la page précédente)

```

${DELAY}                0
${ADMIN_USER}            admin
${ADMIN_PASSWORD}        admin
${PROJECT_URL}           http://${SERVER}/${PROJECT_NAME}/
${PATH_BIN_FILES}        ${EXECDIR}${/}binary_files${/}
${TITLE}                 :: openMairie :: openexemple

*** Keywords ***
For Suite Setup
    # Les keywords définit dans le resources.robot sont prioritaires
    Set Library Search Order    resources
    Ouvrir le navigateur
    Tests Setup

For Suite Teardown
    Fermer le navigateur

```

5.2.11 *tests/resources/app/*

Répertoire contenant les fichiers de déclaration de mots clé dédiés à l'application.

5.2.12 *tests/resources/app/gen/*

Répertoire destiné à recevoir des fichiers de mots clé générés à partir du modèle de données.

5.2.13 *tests/resources/app/__init__.py*

Fichier vide pour définir le répertoire *app* comme un module python.

5.2.14 *tests/resources/app/om_tests.py*

```

#!/usr/bin/python
# -*- coding: utf-8 -*-
from resources.core.om_tests import om_tests_core

class om_tests(om_tests_core):
    """
    """

    _database_name_default = "openexemple"
    _instance_name_default = "openexemple"

```

5.2.15 *tests/resources/app/keywords.robot*

```
*** Settings ***
Documentation    Keywords openexemple.

*** Keywords ***
Depuis le listing
    [Documentation]
    [Arguments]  ${listing_obj}
    Go To       ${PROJECT_URL}scr/tab.php?obj=${listing_obj}
```

5.2.16 tests/resources/core/

Répertoire récupéré depuis le core du framework via un EXTERNALS.

```
tests/resources/core/  svn://scm.adullact.net/svnroot/openmairie/openmairie_exemple/
↪trunk/tests/resources/core/
```

5.3 Fonctionnement et Utilisation

5.3.1 Pré-requis

Les tests doivent être joués dans un environnement balisé et reproductible à l'identique. Pour ce faire il est nécessaire avant chaque lancement de test, de dérouler une routine qui permet de mettre en place un environnement de tests. Un script permet de dérouler cette routine de manière automatisée :

```
./om-tests -c initenv
```

Ce script permet de :

- supprimer la base de données
- créer la base de données
- initialiser la base de données grâce au script data/pgsql/install.sql
- redémarrer apache pour prendre les traductions en compte
- donner les droits à apache pour les dossiers dans lequel il peut écrire
- faire un lien symbolique vers le dossier de l'applicatif pour que les tests en question dans le dossier /var/www/

Les tests sont prévus pour être exécutés sur le navigateur Firefox. Il est possible d'utiliser une version spécifique automatiquement lors de l'exécution des tests. Pour définir une version de navigateur spécifique il faut :

- télécharger le navigateur Firefox conseillé :
 - 64 bits
 - 32 bits
- extraire l'application dans le dossier souhaité
- créer un fichier de configuration dans votre dossier utilisateur :

```
vim ~/.om-tests/config.cfg
[browser]
src_path=[chemin du navigateur spécifique]
dest_path=/usr/local/bin/firefox
```

5.3.2 Tous les tests

Lancer tous les tests avec initialisation de l'environnement de tests

```
./om-tests -c runall
```

5.3.3 Un seul TestSuite

Lancer un TestSuite avec initialisation de l'environnement de tests

```
./om-tests -c runone -t 000_testsuite_a_executer.robot
```

Lancer un TestSuite sans initialisation de l'environnement de tests

```
./om-tests -c runone -t 000_testsuite_a_executer.robot --noinit
```

5.4 Développement et bonnes pratiques

Il est prévu de consigner ici les bonnes pratiques et les consignes pour le développement des tests.

5.4.1 Documentation RobotFramework

Librairie du framework openMairie [Core](#).

Cette documentation de la librairie du framework openMairie a été générée avec la commande suivante :

```
./om-tests -c gendoc
```

La commande est automatiquement exécutée lorsque l'on lance un ou tous les TestSuite. La documentation est générée au format HTML dans le répertoire *tests/doc*. Il y a une documentation par dossier de ressources :

- *tests/resources/app* → *tests/doc/app.html*
- *tests/resources/core* → *tests/doc/core.html*

RobotFramework :

- <http://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html>

Librairies :

- Base - BuiltIn : <http://robotframework.org/robotframework/latest/libraries/BuiltIn.html>
- Base - String : <http://robotframework.org/robotframework/latest/libraries/String.html>
- Base - Collections : <http://robotframework.org/robotframework/latest/libraries/Collections.html>
- Base - OperatingSystem : <http://robotframework.org/robotframework/latest/libraries/OperatingSystem.html>
- Selenium2 : <http://rtomac.github.io/robotframework-selenium2library/doc/Selenium2Library.html>
- Requests : <http://bulkan.github.io/robotframework-requests/>
- Selenium2Screenshots : https://robotframework-selenium2screenshots.readthedocs.org/en/latest/_downloads/keywords.html

5.4.2 Convention de nommage

- Un fichier de test par thème fonctionnel, une TestCase par fonctionnalité.
- **Convention de nommage :**
 - des fichiers : *mon_theme_fonctionnel.robot*
 - des testcase : Saisir un nouvel élément

5.4.3 Génération

Pré-requis : créer le dossier “gen” dans “../tests/resources/core/gen/”.

Lancer une génération complète à chaque modification de la structure de la base de données permet de créer les mots-clefs basiques de chaque table : « depuis le contexte », « ajouter », « modifier », « supprimer » et « saisir ».

5.4.4 Bonnes pratiques

- Éviter d'utiliser les sélecteurs XPATH, les sélecteurs CSS ou par ID sont largement préférables.
- Isolation des tests : chacun des tests ajouté doit être indépendant de ceux existants (consitution de son propre jeu de données, accès aux éléments par recherche, éventuellement nettoyage des données créées, etc).

Historique & Mises à niveau

6.1 La version 4.5

6.1.1 Les nouveautés de la version 4.5

Intégration du WYSIWIG dans la mise en forme des états (bibliothèque js tinyMCE)

Nouveau module SIG (information géographique)

Nouvelle fonctionnalité « custom » permettant de « customizer » son application « métier »

Amélioration de la recherche avancée par l'ajout de filtre

Amélioration du générateur avec de nouvelles possibilités de configuration

Intégration des scripts « scr » dans la classe du générateur afin de pouvoir favoriser la personnalisation par surcharge

Utilisation du filestorage lors de la génération de CSV (avec reqmo par exemple)

Ajout de point spécifique d'entrée dans le framework et ajout de la possibilité d'accès direct à un objet (direct link)

6.1.2 Mettre à niveau depuis openMairie 4.4 vers 4.5

- Le script `dyn/dashboard.inc.php` n'est plus appelé. Il était utilisé pour surcharger le système de tableau de bord standard de l'application. Il est désormais surchargeable via la méthode `om_dashboard::view_dashboard()`.
- Remplacer dans l'application tous les appels au script `scr/requeteur.php` par des appels au script `scr/reqmo.php`. Le code du module "Reqmo" a été factorisé et a entraîné la suppression du script en question.
- Le système de stockage par défaut des fichiers a évolué vers le système "filesystem". Il est fortement conseillé d'envisager la migration des données de l'application vers ce système. Si cette migration n'est pas réalisable dans l'immédiat, l'ancien système est toujours disponible via un fichier de configuration `dyn/filestorage.inc.php` suivant :

```
<?php
$filestorage = array();
$filestorage["filestorage-default"] = array (
    "storage" => "deprecated", // l'attribut storage est obligatoire
    "path" => "../trs/1/", // attention il est peut être nécessaire de remplacer 1_
    ↪ par la clé de la configuration de votre base de données
    "temporary" => array(
        "storage" => "filesystem", // l'attribut storage est obligatoire
        "path" => "../tmp/", // le repertoire de stockage
    ),
);
?>
```

- La fonction javascript `traces()` a été supprimée du framework. Il suffit de copier/coller la définition de cette fonction dans le fichier javascript `../app/js/script.js` de l'application si elle est utilisée.
- La méthode `getPathFolderTrs()` a été supprimée du framework. Il suffit de copier/coller la définition de cette méthode dans la classe `utils` de l'application `../obj/utils.class.php` si elle est utilisée.
- La méthode `application::displayAllScriptJsCall()` a été supprimée du framework. Il suffit de copier/coller la définition de cette méthode dans la classe `utils` de l'application `../obj/utils.class.php` si elle est utilisée.

6.2 La version 4.4

6.2.1 Les nouveautés de la version 4.4

- fonctionnement du générateur sur `pgsql`
- nouveau `filestorage`
- nouveau dashboard : `om dashboard` et `om widget`
- version `sig` (single tile, filtre `wms`, tile cache)
- mode consultation
- abstraction du layout
- état : nouveau `om etat`, `om logo` et `om requête`

6.2.2 Mettre à niveau depuis openMairie 4.3 vers 4.4

Cette procédure permet de mettre à niveau une application utilisant openMairie version 4.3.0 vers openMairie 4.4.0.

Pour conserver une application fonctionnelle tout au long de la mise à niveau, il est vivement conseillé de :

- suivre les étapes de cette procédure dans l'ordre ;
- ne pas utiliser le générateur lorsque ce n'est pas indiqué.

Consultez la section sur les erreurs connues si des erreurs persistent après la mise à niveau.

6.2.2.1 Étape 1 : mettre à jour la base de données

6.2.2.1.1 La structure

La structure de la base de données d'openMairie a changée sensiblement depuis la version 4.3.0. Pour mettre à jour la base de données depuis cette version il faudra exécuter le script SQL `ver_4.4.0.sql`.

Pour PostgreSQL :

`/data/pgsql/ver_4.4.0.sql`

Important : Le support de mysql à été abandonné.

Régénérer les fichiers via le générateur.

Vérifier les surcharges des objets : obj/ et sql/

6.2.2.2 Étape 2 : mise à jour du menu

Suite à l'ajout d'une table contenant les logos et une autre contenant les requêtes, les entrées correspondantes dans le menu devraient être ajoutées.

6.2.2.3 Étape 3 : vérification des requêtes et logos

Vérifier que les logos et requêtes sont dans les bons état/lettres type.

6.2.2.4 Étape 4 : système de stockage des fichiers

Le système de stockage de fichier ayant été mis à jour, le système utilisé conserve la compatibilité avec les fichiers existant.

6.2.2.5 Les erreurs connues

6.3 La version 4.3

6.3.1 Les nouveautés de la version 4.3

Modification om sig

Prise en compte des contraintes clés primaires et étrangère dans le générateur .

Re écriture des actions en tableau : href tab_action

6.3.2 Mettre à niveau depuis openMairie 4.2 vers 4.3

Cette procédure permet de mettre à niveau une application utilisant openMairie version 4.2.0 vers openMairie 4.3.0.

Pour conserver une application fonctionnelle tout au long de la mise à niveau, il est vivement conseillé de :

- suivre les étapes de cette procédure dans l'ordre ;
- ne pas utiliser le générateur lorsque ce n'est pas indiqué.

Consultez la section sur les erreurs connues si des erreurs persistent après la mise à niveau.

6.3.2.1 Étape 1 : mettre à jour les surcharges du framework

6.3.2.1.1 Classe `application`

Supprimer l'utilisation de l'attribut :

```
<?php
var $permission_if_right_does_not_exist = true;
?>
```

S'il est utilisé dans une surcharge, il doit être remplacée par :

```
<?php
$this->config['permission_if_right_does_not_exist'];
?>
```

Les méthodes surchargés de la classe `om_application` doivent être mises à jour avec leur nouvelle implémentation.

6.3.2.1.2 Classe `dbForm`

Les méthodes surchargés de la classe `dbForm` doivent être mises à jour avec leur nouvelle implémentation.

6.3.2.1.3 Classe `formulaire`

Les méthodes surchargés de la classe `formulaire` doivent être mises à jour avec leur nouvelle implémentation.

6.3.2.1.4 Classe `table`

Les méthodes surchargés de la classe `table` doivent être mises à jour avec leur nouvelle implémentation.

Supprimer l'utilisation de la méthode :

```
<?php
function countHrefColumns($href = array())
?>
```

Si elle est utilisée dans une surcharge, elle doit être remplacée par :

```
<?php
function countActions($actions)
?>
```

6.3.2.2 Étape 2 : mettre à jour la base de données

6.3.2.2.1 La structure

La structure de la base de données d'openMairie a changée sensiblement depuis la version 4.2.0. Pour mettre à jour la base de données depuis cette version il faudra exécuter le script SQL `ver_4.3.0.sql`.

Pour MySQL :

```
/data/mysql/ver_4.3.0.sql
```

Pour PostgreSQL :

```
/data/pgsql/ver_4.3.0.sql
```

6.3.2.2.2 Les tables métier

Le générateur gère maintenant plusieurs contraintes :

- PRIMARY KEY
- FOREIGN KEY
- UNIQUE
- NOT NULL

En fonction de ces contraintes les fichiers de l'application sont générés différemment par rapport à openMairie version 4.2.0.

6.3.2.2.2.1 PRIMARY KEY

Ajouter la contrainte SQL PRIMARY KEY.

Le générateur peut maintenant utiliser les clés primaires. Pour créer le champ identifiant, il faudra utiliser la contrainte `PRIMARY KEY` à la place des noms de table en tant que nom de colonne.

6.3.2.2.2.2 FOREIGN KEY (PostgreSQL)

Ajouter la contrainte SQL FOREIGN KEY.

Le générateur gère également les clés étrangères des bases PostgreSQL. Pour créer des références, il faudra utiliser la contrainte `FOREIGN KEY` à la place des noms de table étrangères en tant que nom de colonne.

6.3.2.2.2.3 UNIQUE

- Ajouter la contrainte SQL UNIQUE.
- Mettre à jour les fichiers de surcharge du répertoire `obj/`.

La contrainte `UNIQUE` permet maintenant de gérer automatiquement les champs uniques. Il n'est plus nécessaire de surcharger la méthode `verifier` des modèles pour gérer ce type de champ. Il faudra nettoyer les surcharges de `verifier` en supprimant la vérification manuelle des champs requis et les remplacer par des contraintes `UNIQUE` dans la base de données.

6.3.2.2.4 NOT NULL

- Ajouter la contrainte NOT NULL aux champs requis.
- Supprimer la clause DEFAULT des champs requis.
- Supprimer la contrainte NOT NULL des champs non-requis ou ajouter la clause DEFAULT en fonction du besoin.
- Mettre à jour les fichiers de surcharge du répertoire obj/.
- Générer.

Toutes les colonnes NOT NULL généreront des champs requis. Des champs qui n'étaient pas requis dans la version 4.2.0 peuvent donc l'être dans la version 4.3.0 après une génération. Il faut donc supprimer la contrainte NOT NULL des colonnes qui ne sont pas réellement requises par l'application ou ajouter une valeur par défaut avec la clause DEFAULT.

Concernant les champs requis par l'application. Il n'est plus nécessaire de surcharger la méthode `verifier` des modèles pour gérer ce type de champ. Il faudra nettoyer les surcharges de `verifier` en supprimant la vérification manuelle des champs requis et les remplacer par des contraintes NOT NULL sans clause DEFAULT dans la base de données.

Important : Vous pouvez générer à nouveau l'application à partir d'ici.

6.3.2.3 Étape 3 : mettre à jour les fichiers de surcharge du répertoire `sql/`

6.3.2.3.1 Alias des tables étrangères

Prefixer le nom des colonnes étrangères par l'alias généré dans `gen/sql/`.

Le générateur peut donner à une table étrangère un alias unique. Cela permet d'effectuer plusieurs jointures sur une même table sans avoir d'erreur d'ambiguïté avec les noms des colonnes. Pour cela, dans les fichiers du répertoire `sql/` contenant plusieurs référence vers une même table étrangère, les noms des colonnes provenant de ces tables étrangères devront être préfixés par l'alias adéquat. Cet alias apparaît dans la définition de la variable `$table` dans les fichiers générés du répertoire `gen/sql/`.

6.3.2.3.2 La clause ORDER BY

Supprimer les surcharges de la variable `$tri` = "ORDER BY libelle", ce tri est généré par défaut.

Le générateur crée maintenant une clause SQL ORDER BY pour chaque modèle. Le tri par défaut se fait sur une éventuelle colonne `libelle`. Si elle n'existe pas la deuxième colonne de la table est utilisée, sinon la clé primaire.

Note : Dans le cas où la deuxième colonne d'une table est utilisée comme libellé, si cette colonne est une clé étrangère, alors le tri se fera sur le libellé de la table étrangère.

6.3.2.3.3 Les actions du tableau

6.3.2.3.3.1 Les actions d'openMairie

- Remplacer `$href[0]` par `$tab_actions["corner"]`["ajouter"].
- Remplacer `$href[1]` par `$tab_actions["left"]`["modifier"].

- Remplacer `$href[2]` par `$tab_actions["left"]["supprimer"]`.

Pour surcharger l'action ajouter, il faut maintenant surcharger `$tab_actions['corner']['ajouter']` et non plus `$href[0]` :

```
<?php

$tab_actions['corner']['ajouter'] =
    array('lien' => 'form.php?obj='.$obj.'&action=0',
          'id' => '&adv_id='.$adv_id.'&tricol='.$tricol.'&valide='.
→$valide.'&retour=tab',
          'lib' => '<span class="om-icon om-icon-16 om-icon-fix add-16" title="'._
→('Ajouter').'">_('Ajouter').</span>',
          'rights' => array('list' => array($obj, $obj.'_ajouter'), 'operator' => 'OR
→'),
          'ordre' => 10,);

?>
```

Pour surcharger l'action modifier, il faut maintenant surcharger `$tab_actions['left']['modifier']` et non plus `$href[1]` :

```
<?php

$stab_actions['left']['modifier'] =
    array('lien' => 'form.php?obj='.$obj.'&action=1.'&idx=',
        'id' => '&premier='.$premier.'&adv_id='.$adv_id.'&recherche='.
→$recherche1.'&tricol='.$tricol.'&selectioncol='.$selectioncol.'&valide=
→'.$valide.'&retour=tab',
        'lib' => '<span class="om-icon om-icon-16 om-icon-fix edit-16" title="'._('
→'Modifier').'">_('Modifier').'</span>',
        'rights' => array('list' => array($obj, $obj.'_modifier'), 'operator' => 'OR
→'),
        'ordre' => 20,);

?>
```

Pour surcharger l'action de contenu, il faut maintenant surcharger `$tab_actions['content']` et non plus `$href[1]` :

```
<?php

$tab_actions['content'] = $tab_actions['left']['modifier'];

?>
```

Pour surcharger l'action supprimer, il faut maintenant surcharger `$stab_actions['left']['supprimer']` et non plus `$href[2]` :

```
<?php
$stab_actions['left']['supprimer'] =
    array('lien' => 'form.php?obj='.$obj.'&action=2&idx=',
          'id' => '&premier='.$premier.'&adv_s_id='.$adv_s_id.'&recherche='.
→ $recherche1.'&tricol='.$tricol.'&selectioncol='.$selectioncol.'&valide='
→ '.$valide.'&retour=tab',
          'lib' => '<span class="om-icon om-icon-16 om-icon-fix delete-16" title="'._
→ 'Supprimer'.'>'._('Supprimer').'</span>',
```

(suite sur la page suivante)

(suite de la page précédente)

```

        'rights' => array('list' => array($obj, $obj.'_supprimer'), 'operator' =>
↪ 'OR'),
        'ordre' => 30,);

?>

```

6.3.2.3.2 Les actions personnalisées

Redéfinir les actions avec la nouvelle manière.

Les actions personnalisées doivent être défini selon la nouvelle manière. Exemple :

```

<?php

$tab_actions['left']['edition'] = array(
    'lien' => '../pdf/pdfetat.php?obj=om_collectivite&idx=',
    'id' => '',
    'lib' => '<span class="om-icon om-icon-16 om-icon-fix pdf-16" title="'.__('Edition
↪ ').">'.__('Edition').'</span>',
    'ajax' => false,
    'ordre' => 21,
);

?>

```

6.3.2.3.3 Définition de l'action

La première clé de \$tab_actions permet choisir la position d'affichage :

- corner pour les actions en coin ;
- left pour les actions de gauche.

La seconde clé de \$tab_actions permet de définir la nouvelle action. Cette clé doit être différente de : ajouter, consulter, modifier et supprimer.

Les clés lien, id et lib s'utilise de la même manière qu'avant.

6.3.2.3.4 Définition du mode d'affichage en sous-tableau

La clé ajax permet d'indiquer si l'action doit être affichée en ajax ou non dans les sous-tableaux :

- true, l'action utilisera la fonction ajaxIt () ;
- false, l'action n'utilisera pas la fonction ajaxIt () .

6.3.2.3.5 Définition de l'ordre d'affichage

La clé ordre permet de déterminer l'ordre d'affichage par rapport aux autres actions.

Chaque action dispose d'une valeur numérique permettant de définir sa place au sein d'une position. L'action numéro 1 s'affichera en premier, l'action numéro 10 s'affichera après les actions de numéro inférieur, etc.

Ordre des actions par défaut d'openMairie :

- ajouter à pour ordre 10 dans la position corner ;
- consulter à pour ordre 10 dans la position left.

Si la position `corner` est sélectionnée :

- 9, l'action s'affichera avant l'action `ajouter` ;
- 11, l'action s'affichera après l'action `ajouter`.

Si la position `left` est sélectionnée :

- 9, l'action s'affichera avant l'action `consulter` ;
- 11, l'action s'affichera après l'action `consulter`.

6.3.2.3.6 Définition des droits d'affichage

La clé `rights` permet de définir le ou les droits nécessaire à l'utilisateur pour visualiser cette action. Cette clé est optionnelle. Si `rights` n'existe pas, tous les utilisateurs pourront visualiser cette action s'ils peuvent visualiser le tableau correspondant.

La clé `list` permet de définir le tableau des droits nécessaire.

La clé `operator` permet de définir l'opérateur utilisé pour pour vérifier les droits de la liste `list` :

- OR, l'utilisateur doit avoir au moins un droit ;
- AND, l'utilisateur doit avoir tous les droits.

6.3.2.4 Les erreurs connues

6.4 La version 4.2

6.4.1 Les nouveautés de la version 4.2

- `openmairie` n est plus un composant et se place en `repertoire core`
- nouveau sig interne : multiple geometries, `om_sig_point` devient `om_sig_map` : geometry autre que point
- rajout du wms et geom complémentaire dans `om_sig`

6.4.2 Mettre à niveau depuis openMairie 4.1 vers 4.2

La version 4.2.0 du framework prend en charge plus de fonctionnalités et donne toutes possibilité de surcharge aux applications :

- surcharge des objets générés par le generateur ;
- surcharge des composants de base openMairie stocké dans `core` ;
- surcharge de la présentation de base (dans `img` et `css`), des thèmes (`om-theme`) dans `app/css` `app/img` ;
- surcharge du javascript de base `app/js/script.js`.

6.4.2.1 EXTERNALS.txt

Vider les 9 repertoires concernés avant de lancer `externals`.

Appliquer le fichier `EXTERNALS.txt` d'`openmairie` :

```
core  svn://scm.adullact.net/svnroot/openmairie/openmairie_exemple/tags/4.2.0/core/
spg   svn://scm.adullact.net/svnroot/openmairie/openmairie_exemple/tags/4.2.0/spg/
scr   svn://scm.adullact.net/svnroot/openmairie/openmairie_exemple/tags/4.2.0/scr/
lib   svn://scm.adullact.net/svnroot/openmairie/openmairie_exemple/tags/4.2.0/lib/
css   svn://scm.adullact.net/svnroot/openmairie/openmairie_exemple/tags/4.2.0/css/
js    svn://scm.adullact.net/svnroot/openmairie/openmairie_exemple/tags/4.2.0/js/
img   svn://scm.adullact.net/svnroot/openmairie/openmairie_exemple/tags/4.2.0/img/
```

(suite sur la page suivante)

(suite de la page précédente)

```
pdf    svn://scm.adullact.net/svnroot/openmairie/openmairie_exemple/tags/4.2.0/pdf/
php    svn://scm.adullact.net/svnroot/openmairie/openmairie_exemple/tags/4.2.0/php/

om-theme svn://scm.adullact.net/svnroot/openmairie/externals/om-theme/kinosura/tags/1.
→ 0.0
```

Deux répertoires sont remarquables :

- `core` contient maintenant la librairie openMairie (4.2.0);
- `om_theme` est le theme de l'application.

Les dossiers suivant sont spécifiques à l'application elle même :

- `app`, scripts spécifiques de l'application, noter dans `specific.txt` les spécificités;
- `data`, scripts SQL d'initialisation de la base de données;
- `dyn`, paramétrage de l'application;
- `gen`, fichiers générés de l'application;
- `obj`, surcharge des objets générés dans `gen/obj/`, surcharge du core d'openMairie `om_dbformdyn`, `om_formulaire` (4.2.0), `om_application` (util);
- `sql`, requêtes SQL surchargeant les fichiers de `gen/sql`;
- `tmp`, fichiers temporaires de l'application (ajouter les droits d'écriture pour le serveur web);
- `trs`, fichiers uploadés par l'application (ajouter les droits d'écriture pour le serveur web).

6.4.2.2 Regenerer les tables avec `genfull.php`

Ajouter les droits d'écriture à `www-data` dans `gen` :

```
$ sudo chmod -R 777 gen
```

6.4.2.3 Modifier les paramètres `dyn`

`locales.inc.php` (charset) `include.inc.php` (core)

6.4.2.4 Dans `obj/`

Ajouter `om_table.class.php`, `om_dbform.class.php`, et `om_formulaire.class.php`.

6.4.2.5 Evolution `om_sig_point` vers `om_sig_map`

`om_sig_map` est le nouvel outil SIG d'openMairie.

Ne concerne que PostgreSQL.

Executer le script `data/pgsql/ver4.2.0.sql`.

Regénérer les 4 nouvelles tables. Ajouter les scripts spécifiques dans `/obj` et `/sql/pgsql`.

7.1 Convention de codage

La convention de codage openMairie s'applique à tout le code qui fait partie de la distribution officielle d'openMairie ainsi qu'aux applicatifs de la gamme. La convention de codage permet de conserver un code consistant et de le rendre lisible et maintenable facilement par les développeurs openMairie.

Sommaire

- *Convention de codage*
 - *L'indentation du code*
 - *Longueur maximum d'une ligne*
 - *Encodage des fichiers*
 - *Tags PHP*
 - *Règles typographiques*
 - *Accolades*
 - *Espacement*
 - *Indentation*
 - *Nommage des classes, fonctions, méthodes et variables*
 - *Expressions*
 - *HTML Valide et W3C*
 - *Les commentaires dans le code*
 - *Quand et comment commenter son code ?*
 - *Images*

7.1.1 L'indentation du code

Pour améliorer la lisibilité, il faut utiliser une indentation de 4 espaces et non pas des tabulations. En effet, les éditeurs de texte interprètent différemment les tabulations alors que les espaces sont tous interprétés de la même façon. De plus lors de commit, les historiques des gestionnaires de versions (CVS ou SVN) sont faussés par ces caractères.

7.1.2 Longueur maximum d'une ligne

Il est recommandé que la longueur des lignes ne dépasse pas 80 caractères afin de garder une bonne visibilité lors de l'affichage sur un terminal.

7.1.3 Encodage des fichiers

L'encodage des fichiers doit être UTF-8.

7.1.4 Tags PHP

Il faut utiliser toujours `<?php [...] ?>` pour délimiter du code PHP, et non la version abrégée `<? [...] ?>`. Cela est la méthode la plus portable pour inclure du code PHP sur des systèmes d'exploitations disposant de configurations différentes. Un saut de ligne unique est ajouté après la fermeture de la balise.

Il faut éviter de mixer PHP et HTML.

Préférer :

```
<?php
var $plop = "attention";
printf(
    '<script type="text/javascript">alert(\'%s\');</script>',
    $plop
);
?>
```

à :

```
<?php
var $plop = "attention";
?>
<script type="text/javascript">alert('<?php echo $plop; ?>');</script>
```

7.1.5 Règles typographiques

7.1.5.1 Accolades

Les accolades sont asymétriques :

```
<?php
function ma_fonction() {
}
if ($variable === true) {
}
```

7.1.5.2 Espacement

Pour les opérateurs à deux composantes, l'opérateur est entouré d'espaces :

```
<?php
$a = 2;
$b = $c + 2;
if ($a > 2) {
    $a = -1;
};
$abc = "une" . "concaténation";
```

En cas de parenthèses :

- insérez un espace avant dans les cas suivants :

```
<?php
if ($a == $b) {
    for ($a = 0; $a < $b, $a++) {
        echo $a;
    }

    foreach ($a as $key => $value) {
        switch ($c) {
            case 23:
                break;
            case 4:
                echo $b;
        }
    }
}
```

- mais pas dans les cas suivants :

```
<?php
function azerty($a = null) {
    fopen($a);
```

Espace après une virgule :

```
<?php
function azerty($a = null, $b = "c") {
```

7.1.5.3 Indentation

- tableaux :

```
<?php
$azerty = array(
    "a" => $b,
    "b" => $a,
);
```

- fonctions et méthodes :

- sur une ligne :

```
<?php
function azerty($a = null, $b = "c") {
```

- sur plusieurs lignes (déprécié) :

```
<?php
public function copier_view(
    $enteteTab,
    $validation,
    $maj,
    &$db,
    $postVar,
    $aff,
    $DEBUG = false,
    $idx,
    $premier = 0,
    $recherche = "",
    $tricol = "",
    $idz = "",
    $selectioncol = "",
    $advs_id = "",
    $valide = "",
    $retourformulaire = "",
    $idxformulaire = "",
    $retour = "",
    $actions = array(),
    $extra_parameters = array()
) {
```

Note : Une fonction ou méthode ne devrait pas posséder autant de paramètres.

7.1.5.4 Nommage des classes, fonctions, méthodes et variables

Les mots composant les noms de classes, fonctions, méthodes et variables doivent être séparés par un underscore et en minuscule (exemple : snake_case).

7.1.5.5 Expressions

Le php étant un langage à typage faible il est nécessaire de comparer les retours de fonctions et méthodes de façon *stricte* :

```
<?php
if (isset($a) === true) {
}
```

7.1.6 HTML Valide et W3C

Le Code HTML rendu doit être valide selon les standards du W3C.

7.1.7 Les commentaires dans le code

Tout le code PHP doit être commenté selon les règles de PHPDocumentor <https://www.phpdoc.org/docs/latest/index.html> :

```

<?php
/**
 * Courte description du fichier
 *
 * Description plus détaillée du fichier (si besoin en est)...
 *
 * @package openmairie
 * @version SVN : $Id$
 */

(defined("PATH_OPENMAIRIE") ? "" : define("PATH_OPENMAIRIE", ""));
require_once PATH_OPENMAIRIE."om_debug.inc.php";
(defined("DEBUG") ? "" : define("DEBUG", PRODUCTION_MODE));
require_once PATH_OPENMAIRIE."om_logger.class.php";

/**
 * Définition de la classe edition.
 *
 * Cette classe gère le module 'Édition' du framework openMairie. Ce module
 * permet de gérer les différentes vues pour la génération des éditions PDF.
 */
class edition {

    /**
     * Instance de la classe utils
     * @var resource
     */
    var $f = null;

    /**
     * Comparaison de chaines de caractères.
     *
     * Fonction permettant de comparer les valeurs de l'attribut title
     * des deux tableaux passés en paramètre.
     *
     * @param array $a Premier élément à comparer.
     * @param array $b Second élément à comparer.
     *
     * @return bool
     */
    function sort_by_lower_title($a, $b) {
        if (strtolower($a["title"]) == strtolower($b["title"])) {
            return 0;
        }
        if (strtolower($a["title"]) < strtolower($b["title"])) {
            return -1;
        }
        return 1;
    }
}

?>

```

7.1.7.1 Quand et comment commenter son code ?

L'objectif est de produire du code facilement lisible, qui permet à un développeur débutant de comprendre la logique implémentée. Il faut donc éviter de paraphraser le code, et réserver les commentaires pour tout ce qui n'est pas compréhensible de premier abord, ou qui fait appel à de la logique *métier*.

Par exemple, éviter ce genre de commentaire :

```
<?php
// si $maj est plus grand que 3
if ($maj >= 3) {
    // alors on met $i à zéro
    $i = 0;
}
```

... qui n'amène aucune information.

Le commentaire suivant, par contre, apporte une explication fonctionnelle pertinente :

```
<?php
// Dans le contexte du dossier d'autorisation alors le tableau affiche
// une colonne supplémentaire pour afficher le numéro du dossier
if ($contexte == "dossier_autorisation") {
    $nb_col = 4;
} else {
    $nb_col = 3;
}
```

7.1.8 Images

Les fichiers images ajoutés dans les applications openMairie doivent être au format PNG (Portable Network Graphics). Ce format permet d'obtenir des images de qualité avec des propriétés de transparence.

7.2 Versionnage

Cette convention de numérotation des versions concerne le framework et les applications. Il est convenu de numérotter les versions sur 3 chiffres séparés par des points avec un éventuel suffixe : X.X.X ou X.X.X-XX.

Composantes du numéro de version :

- Le premier chiffre représente une version majeure.
- Le deuxième chiffre représente une évolution mineure.
- Le troisième chiffre représente un correctif de bug.
- Un éventuel suffixe permet de décliner la version (uniquement possible sur une version X.X.0). Les suffixes sont : -aX pour alpha (champ fonctionnel non arrêté), -bX pour beta (champ fonctionnel arrêté - testable par des développeurs/intégrateurs), -rcX pour release-candidate (version presque finale - testable sur des environnements de production choisis). Ces versions ne sont pas maintenues et doivent être utilisées avec précaution (non stables). Sans suffixe la version est considérée comme stable et publiable.

Exemples de versionnage : openElec 4.2.0-a1, openCourrier 4.2.0-b1, openADS 4.2.0-rc1, openMairie 4.2.0.

7.3 Documentation

Ce paragraphe vise à regrouper les bonnes pratiques qui permettent à toute la communauté openMairie de travailler sur les mêmes bases et avec les mêmes références concernant les outils de rédaction et de publication de documentation

afin de faciliter la contribution et d’obtenir un rendu homogène.

Voici les postulats :

- chaque logiciel/application possède une seule documentation Sphinx qui contient un manuel d’utilisation et un guide technique bien séparés
- le code source de la documentation doit être déposé dans un dépôt GIT sur github.com
- la génération de la documentation est gérée de manière automatique par readthedocs.org
- docs.openmairie.org est l’unique interface d’accès pour toutes les documentations
- un lien dans le footer de chaque application permet d’accéder à la version de l’application du manuel d’utilisateur sur le site docs.openmairie.org

7.3.1 Quelques bonnes pratiques

7.3.1.1 Les blocs de code PHP

Lorsqu’un exemple de code PHP est présenté dans la documentation, afin d’obtenir une coloration syntaxique plus claire à la lecture, il doit être déclaré de la manière suivante

```
.. code-block:: php

<?php
...
?>
```

7.4 Publication

7.4.1 La documentation

Lorsqu’il y a une nouvelle version de l’application et que la version majeure ou mineure est incrémentée, il faut ajouter une nouvelle version de la documentation aussi.

Voici la liste des étapes à reproduire :

Sur GitHub :

- ajouter une nouvelle branch en reprenant la version majeure et mineure pour la nommer ;
- dans le readme de la documentation, modifier les versions ;
- dans le fichier source/conf.py, modifier les variables project, copyright, version et release ;
- dans le fichier source/index.rst, modifier le titre de la documentation ;
- dans settings, modifier la branche par défaut pour mettre la nouvelle.

Sur readthedoc :

- dans le menu admin, puis version, changer la version par défaut ;
- désactiver la version stable et latest.

Depuis l’URL docs.openmairie.org faire un « refresh » pour mettre à jour la page de présentation des documentations : <http://docs.openmairie.org/?refresh>

Dans l’application :

- modifier le lien dans les fichiers *dyn/footer.inc.php* et *doc/index.php* pour pointer vers la nouvelle URL.

7.5 Stratégies de développement

On définit ici plusieurs stratégies de développement co-existant au sein de l’écosystème des applications openMairie :

- SD01 - haut niveau de qualité
- SD02 - niveau de qualité intermédiaire
- SD03 - niveau de qualité pauvre (Proof of Concept)

Ces stratégies peuvent proposer des règles de contribution (qui peut contribuer, et de quelle manière), mais ne se substituent pas à celles en vigueur au sein de l'application concernée. Elles sous-entendent également une adhésion complète aux règles et aux bonnes pratiques de développement des applications openMairie présentes dans cette documentation.

7.5.1 SD01

7.5.1.1 Principes

- Le tronc est toujours stable
- Chaque évolution / correction se fait dans une branche
- Chaque évolution / correction est documentée
- Chaque évolution / correction est testée
- Technique de développement en TDD
- Tous les tests de l'application doivent passer

7.5.1.2 TDD (Test-Driven Development)

Le TDD, ou développement piloté par les tests repose sur le principe suivant :

Chaque implémentation d'une nouvelle fonctionnalité ou chaque correction de bug commence par l'écriture de tests.

7.5.1.3 Scénario-type de développement d'une évolution

7.5.1.3.1 Initialisation

L'itération de développement commence par la création d'une branche dédiée sur le gestionnaire de contrôle de version (subversion, git). Cette branche est issue du tronc, et doit être nommée explicitement selon la fonctionnalité implémentée.

7.5.1.3.2 Rédactions des tests

Avant tout développement, des tests modélisant la fonctionnalité à implémenter sont rédigés. Selon le type d'évolution, il peut s'agir de :

- tests fonctionnels (on teste les scénarios utilisateur - Librairie RobotFramework)
- tests d'intégration (comment s'intègre ma fonctionnalité dans les scénarios existants - Librairie RobotFramework)
- tests unitaires (on teste méthodes et fonctions unitairement - Librairie PHPUnit)

Une fois rédigés, les tests sont lancés, et doivent être en échec (évolution non-implémentée).

7.5.1.3.3 Documentation

À ce stade, la nouvelle fonctionnalité doit être documentée dans le manuel utilisateur de l'application concernée, dans une branche ou un fork.

7.5.1.3.4 Implémentation

Par la suite, l'implémentation commence, et se poursuit jusqu'à ce que les tests rédigés initialement passent.

7.5.1.3.5 Intégration

Tous les tests de l'application sont relancés et doivent passer avec succès. Dans le cas où l'un d'entre-eux échoue, il doit être corrigé.

7.5.1.3.6 Incorporation

L'évolution peut alors être incorporée sur le tronc de l'application. Pour ce faire, la branche est ré-actualisée avec les dernières modifications potentielles du tronc. Dans le cas où le tronc a évolué depuis la création de la branche, l'étape d'intégration précédemment décrite est conduite à nouveau. La branche peut ensuite fusionnée dans le tronc.

À ce stade, la branche / le fork de la documentation est également intégré dans la documentation principale.

7.5.1.4 Scénario-type de correction de bug

Le déroulement est similaire à celui décrit ci-dessus. Seule la méthodologie d'écriture des tests diffère :

- Dans le cas où il existe des tests couvrant la fonctionnalité dans laquelle est relevé le bug, ceux-ci sont altérés pour le prendre en compte.
- Dans le cas où il n'existe pas de scénario de tests permettant de matérialiser le bug, un nouveau test dédié est rédigé.

La correction du bug s'effectue alors, jusqu'à ce que les tests altérés ou nouvellement créés passent.

7.5.1.5 Règles de contribution

Il existe 3 rôles dans le cadre du développement d'un projet respectant la SD01 :

- Développeur
- Contributeur
- Responsable de version

Cette section ne couvre pas l'aspect fonctionnel et décisionnel (choix et temporalité de la feuille de route, identification des processus métiers évolutif ou réglementaire, préservation de la cohérence de l'application, etc) qui reste la responsabilité propre de l'instance de pilotage de l'application (Comité de pilotage, communauté, etc).

7.5.1.5.1 Contributeur

7.5.1.5.1.1 Qui ?

Pour contribuer, la première étape consiste à obtenir le droit de commit sur le dépôt du code-source du projet. La procédure est la suivante :

1. Le candidat rejoint le projet (ex : compte sur adullact et demande de participation au projet).
2. Le *Chef de projet* valide la candidature.

7.5.1.5.1.2 Comment contribuer ?

Le *Contributeur* doit respecter les règles de développement de la stratégie courante.

Chaque nouvelle fonctionnalité est développée dans une branche, et est accompagnée de sa documentation et de ses tests. À l'issue du développement, le *Contributeur* signale par le moyen de son choix son souhait d'intégration dans le tronc commun. À ce stade, tous les points présentés dans la check-list ci-dessous doivent avoir été couverts. La branche est ensuite validée puis fusionnée dans le tronc par un *Développeur*.

7.5.1.5.1.3 Checklist

- La fonctionnalité est entièrement documentée - [CTRL_DOC]
- La fonctionnalité est testée (les nouveaux cas d'utilisation sont couverts) - [CTRL_TEST]
- L'intégralité des tests existants passent sur la branche à jour - [CTRL_REGRESSION]

7.5.1.5.2 Développeur

Le *Développeur* peut intégrer sur le tronc un développement fait dans une branche, par lui-même ou un tiers. Il est cependant d'usage d'éviter d'intégrer ses propres évolutions.

7.5.1.5.2.1 Qui ?

Le *Développeur* acquiert sa qualité par l'un des moyens suivants :

- Il est membre fondateur du projet
- Il est *Contributeur*, et actif sur une période suffisamment étendue, pour qu'un ensemble significatif de *Développeurs* puisse constater le respect des règles qui régissent la stratégie SD01. A l'issue de quoi, à sa demande et après validation du *Chef de projet*, il peut devenir *Développeur*.

7.5.1.5.3 Responsable de version

Documentation en cours.

7.5.1.6 Label

Les applications openMairie respectant cette stratégie de développement sont estampillées avec le logo suivant :



7.5.2 SD02

7.5.2.1 Principes

...

7.5.2.2 Label

Les applications openMairie qui suivent cette stratégie de développement sont estampillées avec le logo suivant :



7.5.3 SD03

7.5.3.1 Principes

...

7.5.3.2 Label

Les applications openMairie qui suivent cette stratégie de développement sont estampillées avec le logo suivant :



Cette section vise à rassembler des liens, des informations, des tutoriels sur des outils qui ne font pas partie du Framework mais qui sont utilisés par la communauté :

8.1 Apache Subversion (SVN)

Site officiel du projet SVN

8.1.1 Pré-requis

Installer subversion : <http://subversion.apache.org/packages.html>

Il existe également des outils graphiques comme TortoiseSVN (Windows).

8.1.2 L'arborescence

Voici l'arborescence standard d'un projet versionné sur un SVN :

```
/trunk/  
/tags/  
/branches/
```

- trunk/ : la version en cours de développement.
- tags/ : les différentes versions publiées. Les dossiers dans tags/ sont des copies du dossier trunk/ à un instant précis. Ils permettent de fixer une version pour la publier. Il est interdit d'effectuer une modification dans un de ces dossiers, la bonne méthode étant de faire la modification dans le trunk/ et de faire une nouvelle version dans le dossier tags/.
- branches/ : ...

8.1.3 Les règles d’or

- Ne jamais commiter dans un tag.
- Ne jamais commiter sans message de commit.
- Ne jamais tagger une version qui contient des externals vers un “trunk”.

8.1.4 Les commandes basiques à connaître

Récupérer une copie locale :

```
svn co svn+ssh://nom-du-développeur@scm.adullact.net/openmairie/openmairie_exemple/  
↪trunk  
    openmairie_exemple
```

Mettre à jour sa copie locale :

```
svn up
```

Voir l’état de sa copie locale :

```
svn st
```

Voir la différence entre sa copie locale et le dépôt :

```
svn diff
```

```
svn ci
```

8.1.5 Externals

C’est une propriété sur le dépôt SVN permettant d’importer du code provenant d’un dépôt différent.

Le fichier EXTERNALS.txt :

```
#  
# created by: svn propset svn:externals -F ./EXTERNALS.txt .  
#  
  
core  svn://scm.adullact.net/svnroot/openmairie/openmairie_exemple/tags/4.2.0/core/  
spg   svn://scm.adullact.net/svnroot/openmairie/openmairie_exemple/tags/4.2.0/spg/  
scr   svn://scm.adullact.net/svnroot/openmairie/openmairie_exemple/tags/4.2.0/scr/  
lib   svn://scm.adullact.net/svnroot/openmairie/openmairie_exemple/tags/4.2.0/lib/  
css   svn://scm.adullact.net/svnroot/openmairie/openmairie_exemple/tags/4.2.0/css/  
js    svn://scm.adullact.net/svnroot/openmairie/openmairie_exemple/tags/4.2.0/js/  
img   svn://scm.adullact.net/svnroot/openmairie/openmairie_exemple/tags/4.2.0/img/  
pdf   svn://scm.adullact.net/svnroot/openmairie/openmairie_exemple/tags/4.2.0/pdf/  
php   svn://scm.adullact.net/svnroot/openmairie/openmairie_exemple/tags/4.2.0/php/  
  
om-theme svn://scm.adullact.net/svnroot/openmairie/externals/om-theme/kinosura/tags/1.  
↪0.0
```

Appliquer les propriétés externals :

```
svn propset svn:externals -F ./EXTERNALS.txt .
svn up
svn ci
```

attention le repertoire ne doit pas etre existant (copie de travail verouillée)

8.1.6 Keywords

8.1.7 Les clients graphiques

Il est recommandé de savoir utiliser et d'utiliser subversion en ligne de commande mais il existe quelques clients graphiques qui permettent de réaliser certaines opérations d'une manière plus conviviale.

- Meld
- TortoiseSVN
- ...

8.1.8 Tutoriaux

8.1.8.1 Importer un nouveau projet

Un nouveau projet est une nouvelle application qui se base sur la dernière version taggée d'openmairie_exemple. Ce tutorial contient certains pré-requis comme la création du projet sur la forge, le fait d'avoir un utilisateur avec les droits corrects sur le projet, le fait d'avoir consulter la [dernière version taggée d'openmairie_exemple](#)

On se positionne dans la dossier tmp pour récupérer la dernière version d'openmairie_exemple

```
cd /tmp
svn export --ignore-externals svn://scm.adullact.net/svnroot/openmairie/
openmairie_exemple/tags/<DERNIERE_VERSION_OPENMAIRIE_EXEMPLE>/ openexemple
```

On cré l'arborescence standard sur le dépôt (si elle n'existe pas déjà)

```
svn mkdir svn+ssh://<NOM_DU_DEVELOPPEUR>@scm.adullact.net/scmrepos/svn/<NOUVEAU_
↪PROJET>/trunk
svn mkdir svn+ssh://<NOM_DU_DEVELOPPEUR>@scm.adullact.net/scmrepos/svn/<NOUVEAU_
↪PROJET>/tags
svn mkdir svn+ssh://<NOM_DU_DEVELOPPEUR>@scm.adullact.net/scmrepos/svn/<NOUVEAU_
↪PROJET>/branches
```

On se positionne dans le dossier précédemment importé pour supprimer les répertoires à récupérer en EXTERNALS depuis le framework

```
cd openexemple
rm -rf core/ css/ img/ js/ lib/ pdf/ php/ scr/ spg/ tests/resources/core/
```

On se positionne dans le dossier précédemment importé pour importer sur le dépôt son contenu

```
cd openexemple
svn import . svn+ssh://<NOM_DU_DEVELOPPEUR>@scm.adullact.net/svnroot/<NOUVEAU_PROJET>/
↪trunk
```

On se positionne dans son dossier de développement pour créer la copie locale du projet

```
cd ~/public_html/
svn co svn+ssh://<NOM_DU_DEVELOPPEUR>@scm.adullact.net/scmrepos/svn/<NOUVEAU_PROJET>/
↪trunk
    <NOUVEAU_PROJET>
```

On se positionne dans le dossier php de l'application pour appliquer les externals

```
cd <NOUVEAU_PROJET>
svn propset svn:externals -F ./EXTERNALS.txt .
svn up
svn ci
```

8.1.8.2 Publier une nouvelle version

Ce tutorial contient certains pré-requis comme le fait d'avoir un utilisateur avec les droits corrects sur le projet ou connaître comment incrémenter le numéro de version de l'application à publier.

Avant de publier une application, il faut vérifier que l'EXTERNALS de la librairie openMairie ne pointe pas vers le "trunk". Pour cela

```
less php/EXTERNALS.txt

#
# created by: svn propset svn:externals -F ./EXTERNALS.txt .
#

openmairie svn://scm.adullact.net/svnroot/openmairie/openmairie/trunk/
fpdf svn://scm.adullact.net/svnroot/openmairie/externals/fpdf/tags/1.6-min/
pear http://svn.php.net/repository/pear/pear-core/tags/PEAR-1.9.1/
db http://svn.php.net/repository/pear/packages/DB/tags/RELEASE_1_7_13/
```

Ici on voit que openmairie pointe vers le "trunk". Nous devons d'abord publier la librairie

```
svn cp svn+ssh://<NOM_DU_DEVELOPPEUR>@scm.adullact.net/openmairie/openmairie/trunk
    svn+ssh://<NOM_DU_DEVELOPPEUR>@scm.adullact.net/openmairie/openmairie/tags/
↪<NOUVELLE_VERSION>
```

Le message pourra être : Tag openmairie <NOUVELLE_VERSION>.

Ensuite il faut changer les EXTERNALS.txt. On remplace dans le fichier php/EXTERNALS.txt, le trunk par la nouvelle version

```
vim php/EXTERNALS.txt

#
# created by: svn propset svn:externals -F ./EXTERNALS.txt .
#

openmairie svn://scm.adullact.net/svnroot/openmairie/openmairie/tags/<NOUVELLE_
↪VERSION>/
fpdf svn://scm.adullact.net/svnroot/openmairie/externals/fpdf/tags/1.6-min/
pear http://svn.php.net/repository/pear/pear-core/tags/PEAR-1.9.1/
db http://svn.php.net/repository/pear/packages/DB/tags/RELEASE_1_7_13/
```

Ensuite on applique le nouveau propset externals une fois placé dans le dossier php (Attention de ne pas oublier le « . » dans la commande svn propset)

```
cd php/
svn propset svn:externals -F ./EXTERNALS.txt .
svn up
```

Ici en faisant un svn info sur le dossier openmairie, nous devons obtenir une URL comme ceci

```
svn info openmairie/
URL : svn://scm.adullact.net/svnroot/openmairie/openmairie/tags/<NOUVELLE_VERSION>
```

Si tout est ok nous pouvons valider nos modifications puis passer à la publication de l'application

```
svn ci
```

Ici on fait une copie du "trunk" vers le dossier "tags" de l'application openmairie_exemple

```
svn cp svn+ssh://<NOM_DU_DEVELOPPEUR>@scm.adullact.net/openmairie/openmairie_exemple/
↳trunk
    svn+ssh://<NOM_DU_DEVELOPPEUR>@scm.adullact.net/openmairie/openmairie_exemple/
↳tags/<NOUVELLE_VERSION>
```

8.1.8.3 svn utilisation

il est propose dans ce chapitre de lister quelques commandes utiles en cas de conflit testées en svn

type de fichier

```
A Ajout de nouveaux éléments à la version locale
M éléments modifiés localement (par rapport à la version de SVN) ;
? éléments inconnus de SVN (non présents dans la version de SVN) ;
U pour les éléments modifiés dans SVN (par rapport à la version locale) ;
C pour les éléments différents entre les versions locale et SVN, et qui posent un
↳conflit à régler manuellement.
?D fichier supprimés (a verifier)
```

revert et diff :

```
svn revert nomfichier // remet dans le dernier etat du svn
↳(soit pas del soit pas modifier)
svn diff nomdossier ou nomfichier // affiche les modifications r/r au dernier svn up
```

resolution de conflit

```
svn st
!      C openmairie_exemple/trunk/authors.txt
>     local édition, suppression entrante sur mis à jour

svn revert openmairie_exemple/trunk/authors.txt
'openmairie_exemple/trunk/authors.txt' réinitialisé
```

deplacer un dossier sur le svn -> commande mv

Exemple : on a créé trunk/trunk/dossiers_source

```
D'abord on renomme le premier dossier trunk en dossier branches
> svn mv svn+ssh://fraynaud@scm.adullact.net/scmrepos/svn/openboisson/trunk
    svn+ssh://fraynaud@scm.adullact.net/scmrepos/svn/openboisson/branches
```

(suite sur la page suivante)

(suite de la page précédente)

```
cela fait branches/trunk/dossiers_source

Ensuite on déplace le dossier trunk qui se trouve maintenant dans branches à la_
↳ racine du dépôt
> svn mv svn+ssh://fraynaud@scm.adullact.net/scmrepos/svn/openboisson/branches/trunk
    svn+ssh://fraynaud@scm.adullact.net/scmrepos/svn/openboisson/trunk

cela fait trunk/dossiers_source
```

créer - déplacer (autre exemple) - détruire un repertoire sur svn

```
créer un dossier documentation sur svn depuis une copie locale
svn import documentation svn+ssh://fraynaud@scm.adullact.net/scmrepos/svn/
↳ opencimetiere/documentation

renommer = renommer sur le svn trunk en temp
svn rename svn+ssh://fraynaud@scm.adullact.net/scmrepos/svn/opencimetiere/
↳ documentation/trunk
    svn+ssh://fraynaud@scm.adullact.net/scmrepos/svn/opencimetiere/
↳ documentation/temp

move = déplacer le dossier temp/trunk vers trunk
svn mv svn+ssh://fraynaud@scm.adullact.net/scmrepos/svn/opencimetiere/documentation/
↳ temp/trunk
    svn+ssh://fraynaud@scm.adullact.net/scmrepos/svn/opencimetiere/documentation/
↳ trunk

delete détruire le repertoire temp
svn del svn+ssh://fraynaud@scm.adullact.net/scmrepos/svn/opencimetiere/documentation/
↳ temp
```

Creation d une nouvelle version

```
Copie en tag de la version

svn cp svn+ssh://fraynaud@scm.adullact.net/scmrepos/svn/openboisson/trunk
    svn+ssh://fraynaud@scm.adullact.net/scmrepos/svn/openboisson/tags/1.0.0beta

export dans un repertoire local openmairie_debitboisson_1.0.0beta sans les_
↳ repertoires .svn

svn export svn+ssh://fraynaud@scm.adullact.net/scmrepos/svn/openboisson/tags/1.0.
↳ 0beta
    openmairie_debitboisson_1.0.0beta
```

8.2 GIT

Site officiel du projet GIT

8.3 Meld

Site officiel du projet Meld

exemple d utilisation :

meld openmairie_recensement/ svn.openmairtrunck/trunk

8.4 POEdit

POEdit est un éditeur de traductions de chaînes en plusieurs langues. Les chaînes présentes dans l'interface des applications openMairie sont celles présentes dans le code. Elles ne peuvent pas comporter d'accent.

Les étapes sont les suivantes :

- 1 - avoir dans le code .php les chaînes à traduire, en texte sans accent ni caractères spéciaux
- 2 - préparer les dossiers de traductions dans le dossier de locales
- 3 - avoir préalablement installé et configuré POEdit
- 4 - configurer le projet dans POEdit et effectuer scan du code afin de détecter les chaînes à traduire
- 5 - traduire les chaînes dans POEdit et sauvegarder

8.4.1 Spécification dans le code des chaînes à traduire

Les chaînes peuvent être traduites, soit en français accentué, soit dans d'autres langues. Pour cela il est nécessaires qu'elles soient présentes dans les fichiers.php en respectant la syntaxe suivante

```
``_('Ma chaine a traduire, sans accent')``
```

Toutes les chaînes de caractères correspondant aux noms de tables et de champs sont générées par le générateur et sont ainsi directement disponibles.

8.4.2 Préparation des dossiers de locales

Chaque application openMairie comporte, à la racine, un dossier appelé `locales`. Ce dossier comporte une structure de type

```
locales/
  fr_FR/
    LC_MESSAGES/
      openmairie.po
      openmairie.mo
```

Il est nécessaire de créer un dossier de langue (par exemple `en_US`) avec son sous-dossier `LC_MESSAGES` pour chaque langue supplémentaire.

Le fichier `.po` contient les définitions de traductions ; il peut être modifié au moyen de POEdit ou directement depuis un éditeur de texte simple.

Le fichier `.mo` contient les traductions sous une forme compilée. Il est généré par POEdit automatiquement lors de chaque sauvegarde des traductions.

8.4.3 Installation et configuration de POEdit

8.4.3.1 Installation

POEdit est disponible nativement dans la plupart des systèmes linux. Il est possible de le télécharger depuis le site officiel pour tous Linux, Windows et Mac OSX.

<http://www.poedit.net/download.php>

Sous Linux Ubuntu ou Debian, il faut, en root, exécuter la commande `apt-get install poedit`

8.4.3.2 Gestion de plusieurs langues

Une fois installé, il faut s'assurer que les `locales` (fichiers de définition de langues) sont correctement installés sur le système.

Sous Linux Ubuntu, pour ajouter une locale, il faut ajouter sa définition dans le fichier de pays correspondant (exemple : `/var/lib/locales/supported.d/fr`) puis lancer la commande, en root, `dpkg-reconfigure locales`. Cela ne sera nécessaire que pour ajouter la prise en charge d'une nouvelle langue.

8.4.4 Configuration d'un projet dans POEdit

- Cliquer sur le bouton "Créer un nouveau projet de traduction", même si le projet comporte déjà une traduction,
- Saisir le nom du projet et le chemin complet du dossier contenant le projet (chemin complet permettant d'aller à la racine du logiciel openMairie)
- Valider. Cela crée le projet. Les fichiers `.po` existants sont automatiquement détectés.
- Cliquer alors sur le bouton "Mettre à jour tous les catalogues du projet". Un scan de code est alors effectué par POEdit. Cela va parcourir tous les fichiers `.php` du dossier afin de détecter toutes les chaînes encadrées par `_()`.

Le scan peut comporter des erreurs (en général des accents dans les chaînes à traduire, des accents dans les commentaires du code, des chaînes à traduire vides etc.). Dans ce cas le fichier et la ligne concernée sont indiqués dans le rapport d'erreur. Il est recommandé de corriger l'erreur et recommencer la mise à jour.

Le rapport affiche les chaînes désuètes (celles qui ne figurent plus dans le code) et les nouvelles chaînes. Les chaînes désuètes sont alors commentées dans le fichier `.po` et n'apparaissent plus dans l'interface de traduction.

8.4.5 Traduction des chaînes

Depuis l'écran du projet dans POEdit, double-cliquer sur le fichier de la langue concernée. La liste des chaînes à traduire apparaît alors. Les nouvelles chaînes sont en premier, les chaînes modifiées en second et les autres chaînes ensuite. Il suffit de cliquer sur une chaîne et entrer la traduction dans le bloc de texte inférieur.

A chaque sauvegarde, le fichier est compilé en un fichier `.mo`.

L'affichage de l'application affiche alors les chaînes traduites à la place des libellés originaux.

Attention, sur certaines configurations, un redémarrage du serveur web peut être nécessaire pour que les traductions soient mises à jour.

8.5 Sphinx

Site officiel du projet Sphinx

8.6 Github.com

Pour pouvoir gérer un projet sur ce site, il faut avoir un utilisateur (le bouton « Sign up » de la page d'accueil permet d'obtenir un compte utilisateur très facilement).

L'objectif d'utiliser ce site est donc de faciliter la contribution à la documentation grâce à l'éditeur en ligne et au système de « Pull Request » et de déclencher simplement la régénération des documentations grâce au lien automatique avec readthedocs.org.

8.6.1 Créer un projet

Public(s) concerné(s) : Administrateur de projet openMairie.

Il s'agit ici, de créer un nouveau repository sur github.com dans l'organisation openmairie. Le projet doit s'appeler openlogiciel-documentation (par exemple pour le logiciel openElec « openelec-documentation » et pour le logiciel openRésultat « openresultat-documentation »). Ne rien faire d'autre dans ce repository.

Depuis le tableau de bord de github.com, un clic sur le bouton « + » puis « New repository » en haut à droite de l'écran, à côté du login de l'utilisateur, permet d'accéder au formulaire de création d'un dépôt.

Voici les informations à saisir :

- Owner : sélectionner « openmairie », pour une meilleure lisibilité du projet tous les dépôts doivent être créés dans cette organisation.
- Repository name : le nom doit être logiciel-documentation sans accents sans espaces en minuscules (par exemple pour le logiciel openElec « openelec-documentation » et pour le logiciel openRésultat « openresultat-documentation »).
- Description : pour que le dépôt sorte correctement dans les recherches, il faut saisir « Documentation Logiciel Sphinx » (par exemple pour openCimetière « Documentation openCimetière Sphinx »).
- Public ou Private : sélectionner « Public » puisque les projets openMairie sont publics.
- Initialize this repository with a README : ne pas cocher la case.

Puis il suffit de cliquer sur le bouton « Create repository ».

8.6.2 Importer la documentation depuis un projet subversion de l'adullact

Public(s) concerné(s) : Administrateur de projet openMairie.

Les pré-requis sont :

- le projet doit déjà être créé sur github.com
- être dans un terminal pour saisir les commandes suivantes
- les commandes : svn, svn2git et git doivent être installées

Étape 0 : Modifier et définir les variables utilisées dans les commandes suivantes

```
export OLDPRODUCTNAME="openfoncier"
export NEWPRODUCTNAME="openads"
export ADULLACTUSER="fmichon"
```

Étape 1 : Récupération des logs

```
mkdir -p ${NEWPRODUCTNAME}-documentation/${NEWPRODUCTNAME}-documentation && cd $
↪ ${NEWPRODUCTNAME}-documentation/${NEWPRODUCTNAME}-documentation
svn log -q svn://scm.adullact.net/scmrepos/svn/${OLDPRODUCTNAME}/documentation > ../
↪ LOG
cat ../LOG | awk -F '|' '{sub("^ ", "", $2); sub(" $", "", $2); print $2" = \"$2"
↪ <\"$2\">\"}' | sort -u > ../authors-transform.txt
cat ../authors-transform.txt
```

Étape 2 : MANUEL - Modification des auteurs

```
=> Il faut faire la correspondance entre les utilisateurs de l'adullact et ceux de_
↳github
=> exemple : fmichon = Florent Michon <fmichon@atreal.fr>
=> fmichon = login de l'addulact
=> Florent Michon = nom complet du contributeur
=> fmichon@atreal.fr = mail référencé dans github
vim ../authors-transform.txt
```

Étape 3 : Vérification de l'intervalle

```
export REVS="$(tail -n2 ../LOG|head -n1|awk '{print $1}'|sed "s/r//"):${head -n2 ../
↳LOG|tail -n1|awk '{print $1}'|sed "s/r//")"
echo $REVS
```

Étape 4 : Récupération de tous les commits (très long)

```
svn2git --authors ../authors-transform.txt --revision $REVS -v svn://scm.adullact.net/
↳scmrepos/svn/${OLDPRODUCTNAME}/documentation
```

Étape 5 : Import du code sur github

```
git remote add origin git@github.com:openmairie/${NEWPRODUCTNAME}-documentation.git
git push -u --all
git push --tags
```

Étape 6 : Suppression de l'ancien dépôt de documentation sur l'adullact pour que personne ne committe dessus

```
svn del -m "Déplacement de la documentation vers Github" svn+ssh://${ADULLACTUSER}
↳@scm.adullact.net/scmrepos/svn/${OLDPRODUCTNAME}/documentation/trunk svn+ssh://${
↳{ADULLACTUSER}@scm.adullact.net/scmrepos/svn/${OLDPRODUCTNAME}/documentation/
↳branches
echo "Documentation déplacée vers https://github.com/openmairie/${NEWPRODUCTNAME}-
↳documentation" > ../MOVED-TO-GITHUB.txt
svn import -m "Déplacement de la documentation vers Github" ../MOVED-TO-GITHUB.txt_
↳svn+ssh://${ADULLACTUSER}@scm.adullact.net/scmrepos/svn/${OLDPRODUCTNAME}/
↳documentation/MOVED-TO-GITHUB.txt
```

8.6.3 Faire l'import initial d'un projet sphinx

Public(s) concerné(s) : Administrateur de projet openMairie.

8.6.4 Contribuer à une documentation

Public(s) concerné(s) : Contributeur membre du projet openMairie.

8.7 readthedocs.org

readthedocs.org est un site qui héberge de la documentation, la rendant accessible et facile à trouver. Il est possible d'importer les documentations sur ce site depuis les système de gestion de version tel que Subversion, Git ou d'autres. Ce site permet de gérer la mise à jour automatique des documentations à chaque commit dans ces systèmes de gestion

de version. Le site supporte également le support des versions mais seulement pour Git et non pas pour Subversion à l'heure où cette documentation est rédigée.

L'objectif d'utiliser ce site est donc de ne pas avoir à se soucier de la génération des documentations. C'est Read-TheDocs.org qui s'en occupe et dans tous les formats html, pdf, epub, ...

Pour pouvoir gérer un projet sur ce site, il faut avoir un utilisateur (le bouton « Inscription » en haut à droite de la page d'accueil permet d'obtenir un compte utilisateur très facilement).

8.7.1 Importer un nouveau projet sur RTD

Public(s) concerné(s) : Administrateur de projet openMairie.

Depuis le tableau de bord de readthedocs.org, un clic sur le bouton « Importer », permet d'accéder au formulaire de création d'un projet sphinx existant.

Voici les informations à saisir :

- Nom : le nom du logiciel sans accents sans espaces en minuscules (par exemple : openelec ou openresultat).
- Repo : l'URL de github.com où est stockée le code de la documentation (par exemple : <https://github.com/openmairie/openelec-documentation.git> pour openelec ou <https://github.com/openmairie/openresultat-documentation.git> pour openresultat).
- Type de dépôt : « Git » puisque le dépôt est sur github.com.
- Description : Le nom du logiciel avec accents avec espaces et avec la casse (par exemple : openElec ou open-Résultat).
- Language : « French » puisque la documentation est francophone.
- URL Projet : « <http://www.openmairie.org/> ».
- Canonical URL : laissons vide pour le moment.
- Single version : ne pas cocher la case.
- Etiquettes : « openmairie ».

Puis il suffit de cliquer sur le bouton « Créer ».

Si la création du projet s'est bien passée une version de la documentation a du être générée, celle-ci est disponible en cliquant sur le bouton « Afficher les docs » sur la page du projet nouvellement créé.

8.7.2 Paramétrer une nouvelle version d'un projet existant

Public(s) concerné(s) : Administrateur de projet openMairie.

Par défaut un projet sur readthedocs.org gère uniquement la dernière version de la documentation "latest" en récupérant la branche par défaut de la documentation sur github.com "master".

Il est possible de gérer plusieurs versions de la documentation pour obtenir des URL du style :

- <http://omframework.readthedocs.org/fr/4.2/>
- <http://omframework.readthedocs.org/fr/4.4/>
- <http://omframework.readthedocs.org/fr/latest/>

Chaque version dans readthedocs.org, correspond à une branche dans le dépôt du projet sur github.com.

...

CHAPITRE 9

Contributeurs

(par ordre alphabétique)

- APITUX
- [atReal](#)
- Alain Baldachino
- Jean Christophe Becquet
- Thierry Benita
- Romain Beylerian
- Matthias Broquet
- Laurent Groleau
- Nicolas Haye
- Jean-Yves Madier de Champvermeil
- Nicolas Meucci
- Florent Michon
- Virginie Pihour
- Francois Raynaud
- Sofien Timezouaght

A

`affichepdf()` (méthode formulaire), 160
`afficher()` (méthode formulaire), 162
`afficherChamp()` (méthode formulaire), 162
`afterFormSpecificContent()`, 160
`ajouter()` (méthode dbform), 158

C

`checkbox()` (méthode formulaire), 160
`checkboxnum()` (méthode formulaire), 160
`checkboxstatic()` (méthode formulaire), 160
`cleSecondaire()` (méthode dbform), 159
`comboD()` (méthode formulaire), 161
`comboD2()` (méthode formulaire), 161
`comboG()` (méthode formulaire), 161
`comboG2()` (méthode formulaire), 161
`create()` (méthode filestorage), 192
`create_temporary()` (méthode filestorage), 192

D

`date()` (méthode formulaire), 160
`date2()` (méthode formulaire), 160
`datestatic()` (méthode formulaire), 161
`dbform` (classe de base), 155
`debutBloc()` (méthode formulaire), 162
`debutFieldset()` (méthode formulaire), 162
`delete()` (méthode filestorage), 192
`delete_temporary()` (méthode filestorage), 193

E

`enpied()` (méthode formulaire), 162
`entete()` (méthode formulaire), 162

F

`finBloc()` (méthode formulaire), 162
`finFieldset()` (méthode formulaire), 162
`formSpecificContent()`, 159
`formulaire` (classe de base), 160
`formulaire()` (méthode dbform), 155

G

`geom()` (méthode formulaire), 161
`get()` (méthode filestorage), 192
`get_temporary()` (méthode filestorage), 193

H

`hidden()` (méthode formulaire), 160
`hiddenstatic()` (méthode formulaire), 160
`hiddenstaticdate()` (méthode formulaire), 160
`hiddenstaticnum()` (méthode formulaire), 160
`http()` (méthode formulaire), 160
`httpclick()` (méthode formulaire), 160

I

`init_select()` (méthode dbform), 156

L

`localisation()` (méthode formulaire), 161
`localisation2()` (méthode formulaire), 161

M

`modifier()` (méthode dbform), 158

P

`pagehtml()` (méthode formulaire), 161
`password()` (méthode formulaire), 160

R

`rvb()` (méthode formulaire), 161
`rvb2()` (méthode formulaire), 161

S

`select()` (méthode formulaire), 161
`select_multiple()` (méthode formulaire), 161
`select_multiple_static()` (méthode formulaire), 161
`selectdisabled()` (méthode formulaire), 161
`selecthiddenstatic()` (méthode formulaire), 161

`selectstatic()` (méthode formulaire), 161
`set_form_default_values()` (méthode dbform), 156
`setBloc()` (méthode dbform, formulaire), 156
`setBloc()` (méthode formulaire), 163
`setFieldset()` (méthode formulaire), 156, 163
`setGroupe()` (méthode dbform), 156
`setGroupe()` (méthode formulaire), 163
`setId()` (méthode dbform), 159
`setKeyup()` (méthode formulaire), 163
`setLayout()` (méthode dbform), 156
`setLib()` (méthode dbform), 156
`setLib()` (méthode formulaire), 163
`setMax()` (méthode dbform), 156
`setMax()` (méthode formulaire), 163
`setOnchange()` (méthode dbform), 156
`setOnchange()` (méthode formulaire), 163
`setOnClick()` (méthode dbform), 156
`setOnClick()` (méthode formulaire), 163
`setOnkeyup()` (méthode dbform), 156
`setRegroupe()` (méthode dbform), 156
`setRegroupe()` (méthode formulaire), 163
`setSelect()` (méthode dbform), 156
`setSelect()` (méthode formulaire), 163
`setTaille()` (méthode dbform), 156
`setTaille()` (méthode formulaire), 163
`setType()` (méthode dbform), 156
`setType()` (méthode formulaire), 163
`setVal()` (méthode dbform), 155
`setVal()` (méthode formulaire), 163
`setvalF()` (méthode dbform), 159
`setvalF()` (méthode formulaire), 163
`setValFAjout()` (méthode dbform), 159
`setValsousformulaire()` (méthode dbform), 155
`sousFormSpecificContent()`, 159
`sousformulaire()` (méthode dbform), 155
`statiq()` (méthode formulaire), 160
`supprimer()` (méthode dbform), 158

T

`text()` (méthode formulaire), 160
`textarea()` (méthode formulaire), 161
`textareahiddenstatic()` (méthode formulaire), 161
`textareamulti()` (méthode formulaire), 161
`textdisabled()` (méthode formulaire), 160
`textreadonly()` (méthode formulaire), 160
`triggerajouter()` (méthode dbform), 159
`triggerajouterapres()` (méthode dbform), 159
`triggermodifier()` (méthode dbform), 159
`triggermodifierapres()` (méthode dbform), 159
`triggersupprimer()` (méthode dbform), 159
`triggersupprimerapres()` (méthode dbform), 159

U

`update()` (méthode filestorage), 192
`upload()` (méthode formulaire), 161
`upload2()` (méthode formulaire), 161

V

`verifier()` (méthode dbform), 159
`verifierAjout()` (méthode dbform), 159
`voir()` (méthode formulaire), 161
`voir2()` (méthode formulaire), 161