openMairie Framework Documentation

Version 4.2

openMairie

26 10 2019

Table des matières

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

Avec plus de 30 applications développées pour les collectivités locales accessibles sur le site http://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 à l'adresse mail suivante : contact@openmairie.org

Nous avons conçu openMairie pour fabriquer une maquette le plus en amont possible en s'appuyant sur le modèle de données créé dans la base de données et en intégrant les 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 chapitre « créer une application » qui permet de prendre en main facilement le générateur et le framework openMairie en vous guidant pas à pas dans le mise en place d'une gestion de courrier.

Le chapitre sur le « framework » complète l'exemple ci dessus en vous décrivant le paramètrage, les classes formulaires et éditions du framework. Il a pour but de vous informer de manière complète sur le fonctionnement du framework

Le chapitre consacré au « générateur » décrit dans le détail le fonctionnement de cet outil et de ses assistants. Cet outil permet de fabriquer la maquette.

La version 4.1.0 permet de construire des applications composites (ou mash up) en intégrant des contenus venant d'application externes. Cela permet de construire rapidemment une application à faible coût grâce à la fusion de multiple service internet

Le chapitre consacré a l" « information geographique » décrit dans le détail le fonctionnement SIG interne d'openMairie combinant les API d'internet avec le framework. (La version 4.2.0 améliore l'interface avec l'intégration de web service)

Le chapitre sur les widgets décrit le tableau de bord paramétrable et individualisé par l'utilisateur permettant l'accès à tout type de fonctions internes ou externes.

Enfin ce document rassemble 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.

Bonne lecture !

CHAPITRE 1

Créer une application

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

1.1 Créer la base de données

Vous devez au préalable copier openmairie_exemple dans le repertoire www de votre serveur apache

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

- Créer une base de données appellée « openmairie »
- Créer les tables nécessaires au framework openMairie avec le fichier sql

  data/mysql/init.sql
- Créer les tables necessaires a 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
```

- modifier le paramétrage openMairie pour faire un accès à la base créée si votre base a un nom différent d'openMairie :

dyn/database.inc.php
voir framework/parametrage

— accéder avec votre navigateur sur openmairie_exemple

```
login : demo
mot de passe : demo
```

Le script mysql de création de la base de l'exemple est le suivant

```
--
-- Structure de la table 'courrier'
--

CREATE TABLE courrier (
  courrier int(8) NOT NULL,
  dateenvoi date NOT NULL,
  objetcourrier text NOT NULL,
  emetteur int(8) NOT NULL,
  service int(8) NOT NULL,
  PRIMARY KEY  (courrier)
) TYPE=MyISAM;

--
-- Structure de la table 'emetteur'
--

CREATE TABLE emetteur (
  emetteur int(8) NOT NULL,
  nom varchar(20) NOT NULL,
  prenom varchar(20) NOT NULL,
  PRIMARY KEY  (emetteur)
) TYPE=MyISAM;

--
-- Structure de la table 'service'
--

CREATE TABLE service (
  service int(8) NOT NULL,
  libelle varchar(20) NOT NULL,
  PRIMARY KEY  (service)
) TYPE=MyISAM;
```

1.2 Créer les formulaires

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

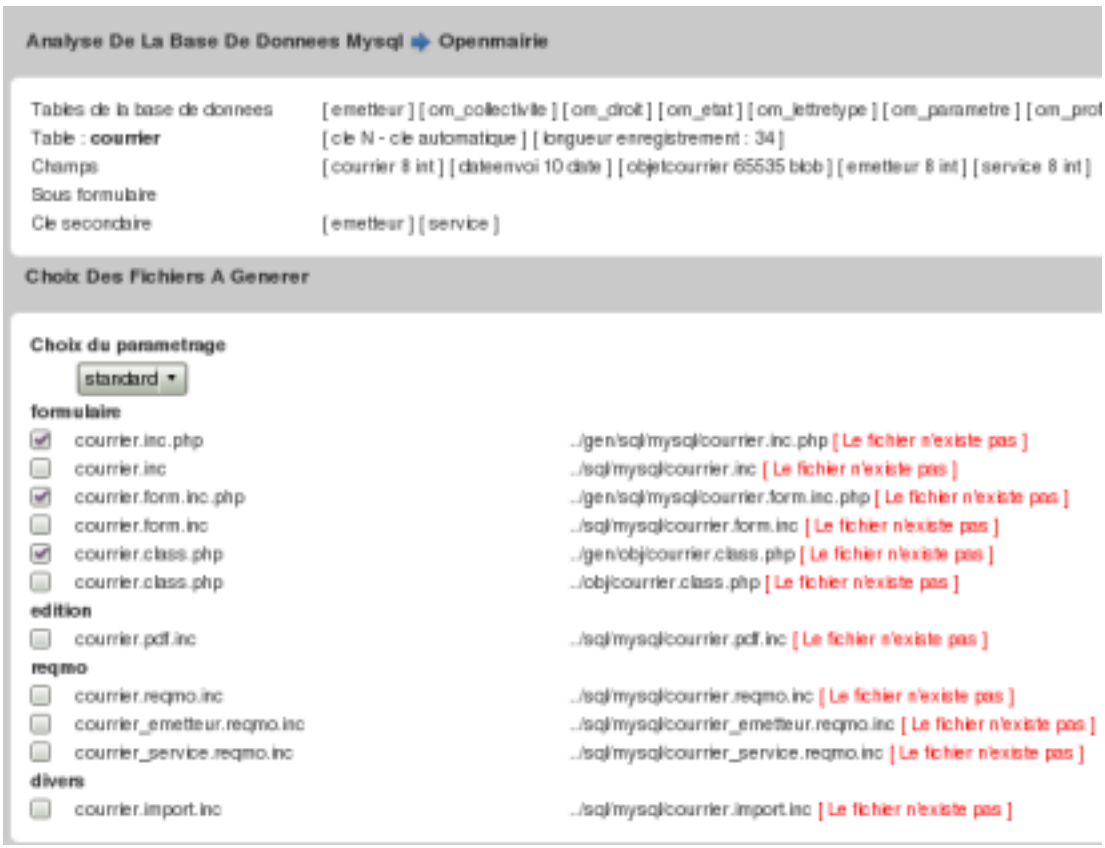
Pour cela, il faut aller dans le menu administration -> generateur

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

Avant de commencer, l'utilisateur apache (www-data) doit avoir les droits d'écriture dans les repertoires /gen , /sql et /obj

1.2.1 Générer les formulaires et édition du courrier

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



Au préalable, le générateur fait une analyse de la base de données (le choix de paramétrage a été supprimé dans la version 4.2.0 - voir paramétrage generateur)

```
Tables de la base de donnees
            [ emetteur ] [ service ] et les tables om..
```

(suite sur la page suivante)

(suite de la page précédente)

```
Table :
            courrier
            [ cle N - cle automatique ]
            [ longueur enregistrement : 34 ]

Champs
            [ courrier 8 int ]
            [ dateenvoi 10 date ]
            [ objetcourrier 65535 blob ]
            [ emetteur 8 int ]
            [ service 8 int ]
Sous formulaire
Cle secondaire
            [ emetteur ] [ service ]
```

Le générateur a détecté 2 clés secondaires et aucun sous formulaire

C'est pour cela qu'il propose 3 « reqmo » : 1 « reqmo » global et 2 « reqmos » suivant la clé secondaire

Par défaut, 3 options sont cochées, ce sont les 3 fichiers fabriqués par le générateur

Cochez toutes les options

```
formulaire
    courrier.inc.php                ../gen/sql/mysql/courrier.inc.php
    courrier.inc                    ../sql/mysql/courrier.inc
    courrier.form.inc.php           ../gen/sql/mysql/courrier.form.inc.php
    courrier.form.inc               ../sql/mysql/courrier.form.inc
    courrier.class.php              ../gen/obj/courrier.class.php
    courrier.class.php              ../obj/courrier.class.php
edition
    courrier.pdf.inc                ../sql/mysql/courrier.pdf.inc
reqmo
    courrier.reqmo.inc              ../sql/mysql/courrier.reqmo.inc
    courrier_emetteur.reqmo.inc ../sql/mysql/courrier_emetteur.reqmo.inc
    courrier_service.reqmo.inc      ../sql/mysql/courrier_service.reqmo.inc
divers
    courrier.import.inc         ../sql/mysql/courrier.import.inc
```

En cliquant sur valider, vous avez le message

```
Parametrage utilise : standard

* ecriture fichier ../gen/sql/mysql/courrier.inc.php
* ecriture fichier ../sql/mysql/courrier.inc
* ecriture fichier ../gen/sql/mysql/courrier.form.inc.php
* ecriture fichier ../sql/mysql/courrier.form.inc
* ecriture fichier ../gen/obj/courrier.class.php
* ecriture fichier ../obj/courrier.class.php
->affichage colone ok 8,23529411765 >= 2.5
* ecriture fichier ../sql/mysql/courrier.pdf.inc
* ecriture fichier ../sql/mysql/courrier.reqmo.inc
* ecriture fichier ../sql/mysql/courrier_emetteur.reqmo.inc
* ecriture fichier ../sql/mysql/courrier_service.reqmo.inc
* ecriture fichier ../sql/mysql/courrier.import.inc
```

Le paramétrage utilisé est le paramétrage standard.

Vous pouvez le modifier : *voir generateur/parametrage*

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)

1.2.2 Générer les formulaires et édition de l'emetteur

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

L'analyse de la base de données est la suivante

```
Tables de la base de donnees
                [ courrier ] [ service ] et les tables om ...

Table :
                emetteur
                [ cle N - cle automatique ]
                [ longueur enregistrement : 48 ]

Champs
                [ emetteur 8 int ]
                [ nom 20 string ]
                [ prenom 20 string ]

Sous formulaire
                [ courrier ]

Cle secondaire
```

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

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

```
Parametrage utilise : standard

* ecriture fichier ../gen/sql/mysql/emetteur.inc.php
* ecriture fichier ../sql/mysql/emetteur.inc
* ecriture fichier ../gen/sql/mysql/emetteur.form.inc.php
* ecriture fichier ../sql/mysql/emetteur.form.inc
* ecriture fichier ../gen/obj/emetteur.class.php
* ecriture fichier ../obj/emetteur.class.php
->affichage colone ok 5,83333333333 >= 2.5
* ecriture fichier ../sql/mysql/emetteur.pdf.inc
* ecriture fichier ../sql/mysql/emetteur.reqmo.inc
* ecriture fichier ../sql/mysql/emetteur.import.inc
```

1.2.3 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

```
Tables de la base de donnees
            [ courrier ] [ emetteur ] et les tables om ..
```

(suite sur la page suivante)

(suite de la page précédente)

```
Table :
        service
        [ cle N - cle automatique ] [ longueur enregistrement : 28 ]

Champs
        [ service 8 int ]
        [ libelle 20 string ]

Sous formulaire
        [ courrier ]

Cle 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

```
Parametrage utilise : standard

* ecriture fichier ../gen/sql/mysql/service.inc.php
* ecriture fichier ../sql/mysql/service.inc
* ecriture fichier ../gen/sql/mysql/service.form.inc.php
* ecriture fichier ../sql/mysql/service.form.inc
* ecriture fichier ../gen/obj/service.class.php
* ecriture fichier ../obj/service.class.php
->affichage colone ok 10 >= 2.5
* ecriture fichier ../sql/mysql/service.pdf.inc
* ecriture fichier ../sql/mysql/service.reqmo.inc
* ecriture fichier ../sql/mysql/service.import.inc
```

1.2.4 Integrer les formulaires dans le menu

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

Nous allons appeller le formulaire depuis le menu :

option application -> tab.php ?obj=courrier

option parametrage -> tab.php ?obj=emetteur

option parametrage -> tab.php ?obj=service

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

```
       // *** APPLICATION ***
       // inserez ici les tables de votre application
         array_push($links,
           array(
               "href" => "../scr/tab.php?obj=courrier",
               "class" => "courrier",
               "title" => _("courrier"),
               "right" => "courrier"
           ));
```

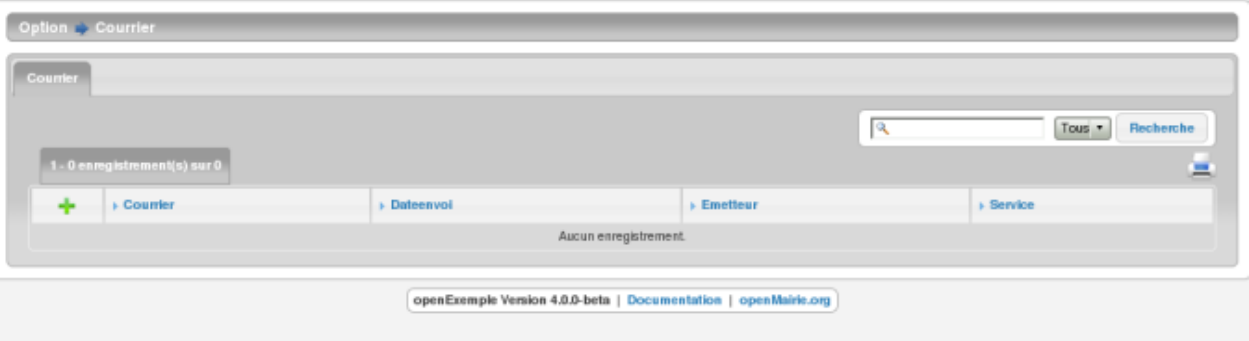
(suite sur la page suivante)

(suite de la page précédente)

```
   // *** TABLES DE PARAMETRAGE ***
   // inserer ici vos tables de parametres

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

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


Vous pouvez accéder à vos formulaires par le menu avec les options :
```

application -> courrier

Cette opération affiche la table courrier :

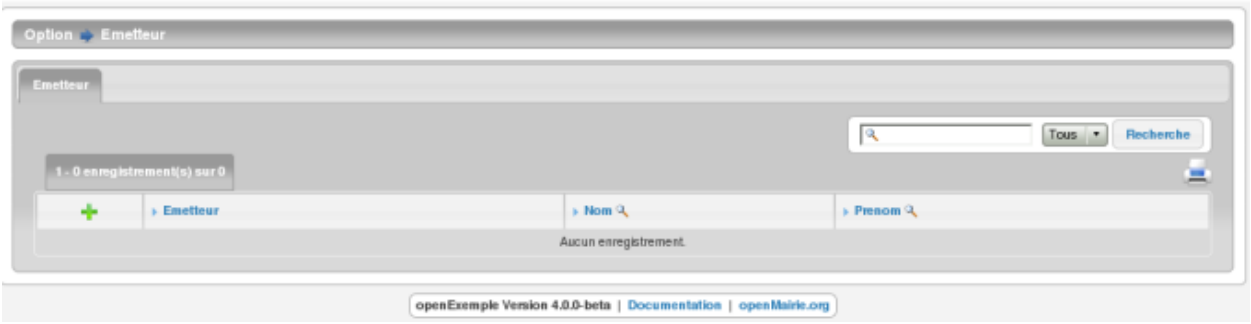


On accéde en appuyant sur + au formulaire d'insertion ou les champs sont :

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

parametrage -> emetteur

Cette operation affiche la table emetteur :

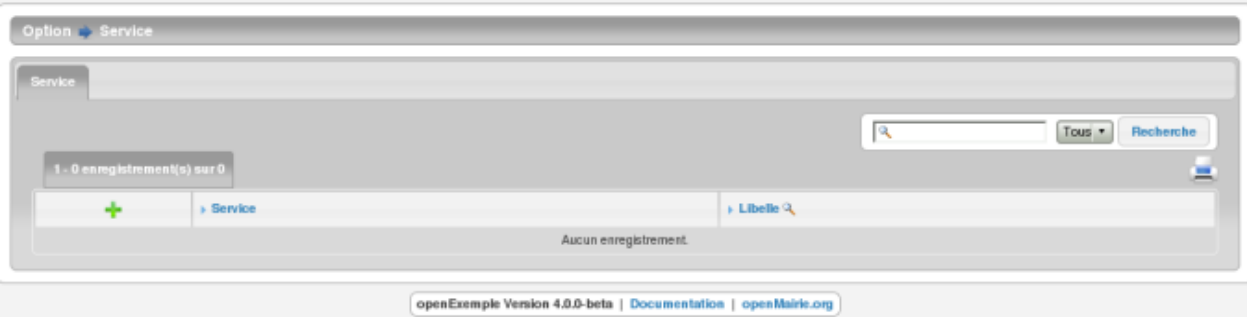


En appuyant sur +, on accède à la saisie

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

parametrage -> service

Cette opération affiche la table service :



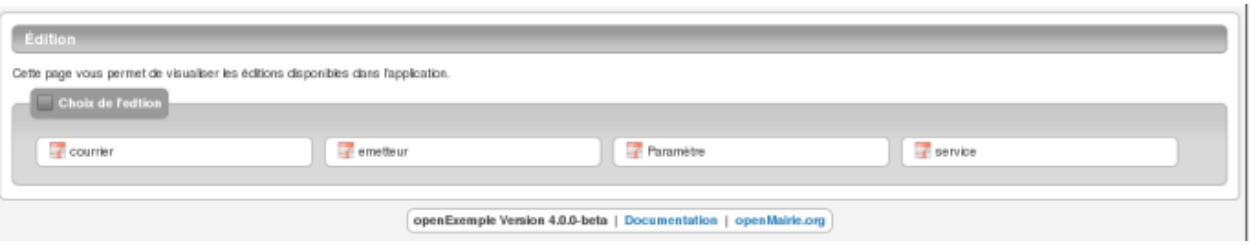
En appuyant sur +, on accede à la saisie

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

Vous pouvez accéder aux éditions et requêtes mémorisées :

export -> edition

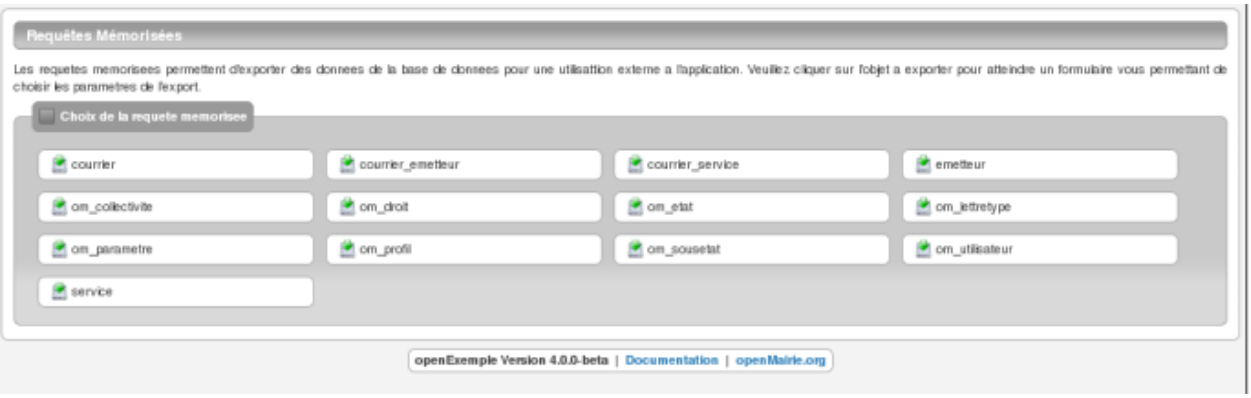
Cet option affiche l'ensemble des éditions pdf :



pour en savoir plus voir *framework/edition*

export -> reqmo

Cette option affiche les requêtes mémorisées :



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 :

pour en savoir plus voir *framework/import*

1.3 Personnaliser son application

Nous allons maintenant personnaliser notre application

Pour se faire, nous allons saisir le jeu de données suivantes

Vous pouvez le faire avec les formulaires, la création des tables stockant les sequences est fait par le framework (methode setId des objets metier) sinon vous pouvez exécuter le script sql suivant

```
-- insertion de deux emetteurs

INSERT INTO emetteur (emetteur, nom, prenom) VALUES
(1, 'dupont', 'pierre'),
(2, 'durant', 'jacques');

--
-- Structure de la table 'emetteur_seq'
--

CREATE TABLE emetteur_seq (
  id int(10) unsigned NOT NULL auto_increment,
  PRIMARY KEY  (id)
) TYPE=MyISAM ;

--
-- Contenu de la table 'emetteur_seq'
--

INSERT INTO emetteur_seq (id) VALUES (2);

--
-- Contenu de la table 'service'
```

(suite sur la page suivante)

(suite de la page précédente)

```
--

INSERT INTO service (service, libelle) VALUES
(1, 'informatique'),
(2, 'telephonie');


--
-- Structure de la table 'service_seq'
--

CREATE TABLE service_seq (
  id int(10) unsigned NOT NULL auto_increment,
  PRIMARY KEY  (id)
) TYPE=MyISAM ;

--
-- Contenu de la table 'service_seq'
--

INSERT INTO service_seq (id) VALUES (2);

--
-- Contenu de la table 'courrier'
--

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

--
-- Structure de la table 'service_seq'
--

CREATE TABLE service_seq (
  id int(10) unsigned NOT NULL auto_increment,
  PRIMARY KEY  (id)
) TYPE=MyISAM ;

--
-- Contenu de la table 'service_seq'
--

INSERT INTO service_seq (id) VALUES (2);
```

1.3.1 Faire un affichage courrier plus convivial

L'affichage des courriers se fait avec les clés secondaires et non les libellés.

Nous souhaitons avoir le nom et le prénom de l'emetteur et le libellé du service.

Dans le fichier sql/mysql/courrier.inc nous allons modifier les variables $table et $champAffiche de la manière suivante (après la ligne include) :

```
$table = DB_PREFIXE."courrier  inner join ".DB_PREFIXE."emetteur
                on emetteur.emetteur=courrier.emetteur
```

(suite sur la page suivante)

(suite de la page précédente)

```
                inner join ".DB_PREFIXE."service
                on service.service=courrier.service";

$champAffiche=array('courrier',
                'concat(substring(dateenvoi,9,2),\'/\',substring(dateenvoi,6,2),
                                \'/\',substring(dateenvoi,1,4)) as dateenvoi',
                'concat(emetteur.nom,\' \',emetteur.prenom) as emetteur',
                'service.libelle as service');
```

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 nous souhaitons rechercher dans les courriers sur le nom de l'emetteur et sur le libellé du service. Dans le fichier sql/mysql/courrier.inc, nous allons modifier la variable tableau $champRecherche de la manière suivante

```
$champRecherche=array("emetteur.nom", "service.libelle");
```

Vous devez avoir dans la zone recherche la possibilité de selectionner

```
Tous
emetteur.nom
service.libelle
```

Nous souhaitons maintenant avoir les derniers courriers au début de la page affichée et nous pouvons le faire en insérant la variable $tri dans courrier.inc de la manière suivante :

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

Le résultat est le suivant

```
2       02/12/2010      durant jacques          informatique
1       01/12/2010          dupont pierre           informatique
```

Pour en savoir plus sur ces variables voir framework/affichage

1.3.2 Rendre obligatoire des champs

Nous avons affiché le courrier avec une jointure de type « inner ». Donc s'il n'y a pas de lien sur le service et/ou l'emetteur, l'enregistrement n'apparaitra pas. Il faut rendre obligatoire la saisie de l'emetteur et du service (auquel le courrier est affecté)

Nous allons surcharger la méthode verifier() dans obj/courrier.class.php de la manière suivante (par défaut le premier champ, ici dateenvoi est obligatoire, cet option est modifiable dans le générateur)

La methode à insérer apres le constructeur est la suivante

```
function verifier($val,&$db,$DEBUG) {
    parent::verifier($val,$db,$DEBUG);
    $f=" !    ";
    $imgv="<img src='../img/punaise.png' style='vertical-align:middle' hspace='2'␣
↪border='0'>";
```

(suite sur la page suivante)

(suite de la page précédente)

```
    if ($this->valF['service']==""){
        $this->msg= $this->msg.$imgv._('service')." "._('obligatoire').$f;
        $this->correct=False;
    }
    if ($this->valF['emetteur']==""){
        $this->msg= $this->msg.$imgv._('emetteur')." "._('obligatoire').$f;
        $this->correct=False;
    }
}
```

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'information voir le châpitre framework/methode

1.3.3 Valoriser un champ par défaut

Pour simplifier la saisie, nous souhaitons mettre la date du jour dans le champ dateenvoi n 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 contient la date systeme (date("Y_m-d")) si la validation est égal à 0 et si $maj est égal à 0 (ajout).

1.3.4 Mettre en majuscule un champ

Nous souhaitons maintenant mettre en majuscule le champ « nom » de la table emetteur. Nous allons surcharger la methode 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 met en majuscule.

1.3.5 Principe à retenir

Voila quelques exemples des possibilités de modification dans les fichiers sql (repertoire sql/ . . . .) et dans les methodes de l'objet (repertoire obj/ . . .)

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

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

1.4 Modifier la base et re générer

Le framework openMairie permet de modifier la base , et de prendre en compte ces modifications en regénérant 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.

1.4.1 Rajouter un champ registre dans courrier

Il est proposer 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_ordre

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

```
ALTER TABLE courrier ADD registre VARCHAR( 20 ) NOT NULL ;
```

Vous devez regénérer votre application courrier dans l'option du menu : administration -> generateur -> courrier et laisser cochées les options par défaut :

gen/obj/courrier.class.php
gen/sql/mysql/courrier.inc.php
gen/sql/mysql/courrier.form.inc.php

Validez l'opération.

Vous pouvez remarquer si vous allez sur le formulaire un nouveau champ registre en fin de formulaire. Votre personnalisation n'est pas affectée.

Nous voulons que le numero 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 a jour de la séquence avant l ajout de l enregistrement

function triggerajouter($id,&$db,$val,$DEBUG)
{
    //  prochain numero de registre
    //  fonction DB pear
    $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 le tableau champAffiche de sql/mysql/courrier.inc de la manière suivante

```
$champAffiche=array('courrier',
    'concat(substring(dateenvoi,9,2),\'/\',substring(dateenvoi,6,2),\'/\',
↪substring(dateenvoi,1,4)) as dateenvoi',
    'concat(emetteur.nom,\' \',emetteur.prenom) as emetteur',
```

(suite sur la page suivante)

(suite de la page précédente)

```
    'service.libelle as service',
    'registre');
```

Votre affichage de la table courrier est modifié.

1.4.2 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. Le script sql est le suivant

```
ALTER TABLE emetteur ADD adresse VARCHAR( 40 ) NOT NULL ,
ADD cp VARCHAR( 5 ) NOT NULL ,
ADD ville VARCHAR( 40 ) NOT NULL ;
```

Vous devez regénérer 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/mysql/emetteur.inc.php
gen/sql/mysql/emetteur.form.inc.php

Validez l'opération.

N'ayant pas modifié sql/mysql/emetteur.inc, le framework fonctionne avec le code généré

1.4.3 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 setGroupe(&$form, $maj) {
    $form->setGroupe('nom','D');
    $form->setGroupe('prenom','F');

    $form->setGroupe('cp','D');
    $form->setGroupe('ville','F');
}

function setRegroupe(&$form,$maj){
    $form->setRegroupe('nom','D', _('nom'), "collapsible");
    $form->setRegroupe('prenom','F','');

    $form->setRegroupe('adresse','D', ('adresse'), "startClosed");
    $form->setRegroupe('cp','G','');
    $form->setRegroupe('ville','F','');
}
```

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

Vos formulaires sont maintenant au point.

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

1.4.4 Les surcharges d'openCourrier

Vous pouvez utiliser openCourrier version 3.0.0 qui est téléchargeable au lien suivant :

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

La base de données d'openCourrier est plus complexe. C'est ainsi que courrier a deux sous formulaires : tache et dossier et qu'il est aussi possible de compléter l'objet du courrier avec une bible.

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

Les méthodes setLib, setGroupe et setRegroupe permettent **une présentation en fieldset** du courrier (utilisation des champs vide1 à 5 voir sql/mysql/courrier.form.inc)

La gestion des emetteurs enregistre dans la table courrier l'emetteur (voir la méthode setType qui utilise les combos, la méthode setSelect qui les paramétre et la méthode triggerAjouterapres qui enregistre l'emetteur saisi en formulaire courrier dans la table emetteur si la case vide5 est cochée)

Il est possible d"**afficher un courrier préalablement scanné** et d" **enregistrer le fichier pdf dans dossier.class.php** après avoir écrit dessus le numéro de registre (Voir les méthodes setType et triggerAjouterapres).

Il y a d'autres objet métier qui ont des surcharges intéressantes :

Dans dossier.class.php, vous avez un exemple de type upload pour télécharger des fichiers.

L'objet obj/tachenonsolde.class.php est un **exemple de surcharge de tache.class.php** qui affiche que les tâches non soldées

openCourrier fonctionne avec des restrictions d'accès par service et **les méthodes de login** ont été modifiées dans obj/utils.class.php ainsi qu" utilisateur.class.php qui a dans openCourrier un champ service.

Vous pouvez aussi regarder **deux scripts de traitement** :
- trt/num_registre.php qui remet à 0 le numéro de registre
- trt/archivage.php qui tranfere en archive les courriers avant une date

Vous avez plus de détail sur les traitements dans le chapitre *framework/util* notament sur la mise à jour du registre.

1.5 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

1.5.1 Créer l'état service

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

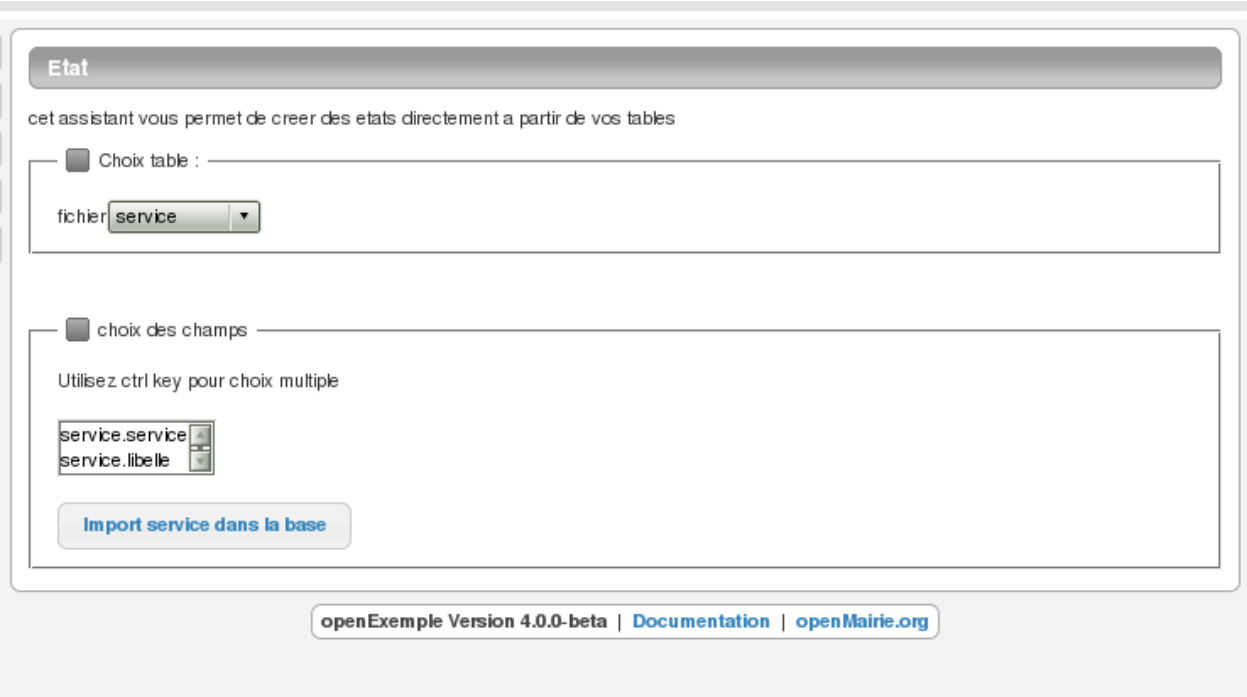
administration -> générateur : assistant

Choisir « créer un état »

Puis choisr dans le select, l'option service

Ensuite avec la touche « ctrl », sélectionner les champs service et libellé

Appuyer ensuite sur « import service dans la base »



Un message apparait « service enregistré »

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

Vous devez rendre d'abord votre etat service « actif » pour pouvoir y accéder.

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

Ouvrer le fichier sql/mysql/service.inc

Ajouter le script suivant

```
$href[3] = array(
    "lien" => "../pdf/pdfetat.php?obj=".$obj."&amp;idx=",
    "id" => "",
    "lib" => "<img src=\"../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 . . .

1.5.2 Créer le sous état courrier

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

administration -> générateur : assistant : sousetat

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

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

En cliquant sur « import courrier dans la base », vous créez un enregistrement ayant pour identifiant « courrier.service » dans la table om_sousetat

1.5.3 Associer le sous état « courrier » à l'état « service »

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

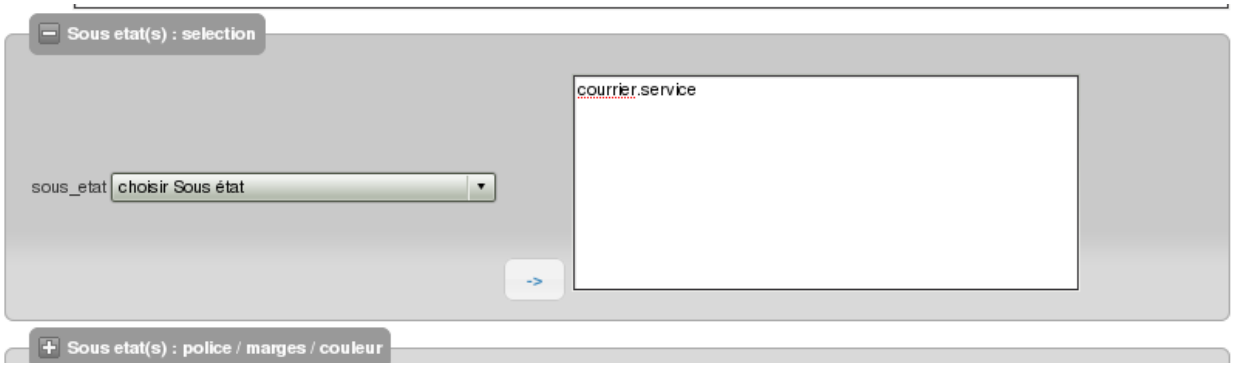
Allez dans l option du menu : administration -> sous etat

Recherchez le sous état « courrier.service » et modifier le en cochant sur actif (1er fieldset)

Il vous faut maintenant associer le sous état « courrier.service » à l'état « service »

Allez dans l'option du menu administration -> etat.

Cherchez l'état « courrier » et modifiez le dans le fieldset (à déplier) sous état selection, choisissez le sous état « courrier.service »



Vous avez désormais un état des courriers par service :

le 27/12/2010

1
informatique

liste courrier

DATEENVOI	OBJETCOURRIER	EMETTEUR	REGISTRE
2010-12-01	Proposition de fourniture de service	1	
2010-12-02	Envoi de devis pour formation openMairie	2	
2010-12-25	essai registre	1	2010-1

1.5.4 Mettre le nom et le prénom de l'emetteur dans le sous état

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

Vous devez modifier la requête sql de l'enregistrement 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 inner join &DB_PREFIXEemetteur
        on emetteur.emetteur = courrier.emetteur
        where courrier.service='&idx'
```

Votre nouvel état a la forme suivante :

le 27/12/2010

1
informatique

liste courrier

DATEENVOI	OBJETCOURRIER	EMETTEUR	REGISTRE
2010-12-01	Proposition de fourniture de service	DUPONTI pierre	
2010-12-02	Envoi de devis pour formation openMairie	durant jacques	
2010-12-25	essai registre	DUPONTI pierre	2010-1

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 etats 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.

CHAPITRE 2

Le framework

openMairie_exemple est le framework de base dans lequel vous pouvez développer votre propre application.

openMairie_exemple est **téléchargeable sur le site de l'adullact**

http://adullact.net/frs/?group_id=329

Il est proposé ici de décrire le fonctionnement du framework.

« 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

Dans un environnement LAMP/WAMP, le framework openMairie integre des composants au travers de classes qui permettent de créer des formulaires et des états. Ces classes sont surchargées par les objets métier à créer.

openMairie integre de nombreux composants : DBPEAR et FPDF dans toutes les applications, mais aussi ARTISHOW (pour les graphes), JQUERY pour l'ergonomie, OPENLAYERS pour l interface SIG . . .

DBPEAR est un abstracteur de base de données qui permet d'utiliser diverses bases de données notament MYSQL ou POSTGRESQL.

FPDF est le composant qui permet de gérer le PDF.

Le développement consiste à créer des objets métier qui surchargent la classe abstraite om_dbformdyn.class.php (composant openMairie). De base, les données de la base de données sont récupérées pour le formulaire (longueur, max, nom).

- om_dbformdyn.class.php ; assure la liaison entre le formulaire et la base de données
- om_formulairedyn.class.php : rassemble toutes les méthodes permettant de construire des formulaires

En introduction, il est proposé une explication de la hiérarchie des répertoires

```
- app : contient tout ce qui est spécifique de votre application avec les
javacripts (app/js/script.js) et les images (app/img)

- css : contient les feuilles de style de base du framework

- core : classe openMairie (version 4.2.0)
```

(suite sur la page suivante)

(suite de la page précédente)

```
- data : contient les requêtes sql permettant de créer votre application
       pgsql : pour postgresql
       mysql pour mysql
   dans les 2 cas, init.sql permet de créer les tables om_xx du framework
                   init_metier.sql sont les tables spécifiques de votre application
   d'autres scripts complètent la mise en oeuvre des données notament les
   modifications de base de données.

- dyn : contient les fichiers de paramètrage du framework décrit ci dessous

- gen : contient les scripts (obj) et requetes (sql) générées par le générateur

- img : contient les images du framework

- js : contient les javascripts de base du framework

- lib : contient les librairies javascripts : openLayers et jquery

- locales : contient les traductions de l'application

- obj : contient les objets métiers surchargeant les objets générés (gen/obj)

- pdf : contient les scripts d'édition du framework

- php : contient les librairies php utilisées par le framework notament : dbpear,␣
↪phpmailer et fpdf

- scr : contient les scripts d'affichage du framework

- spg : contient les sous programmes génériques du framework

- sql : contient les scripts sql surchargeant les scripts générés (gen/sql ...)

- tmp : contient les fichiers temporaires créés par l'application

- trs : contient les fichiers uploadés de l'application (par base /1 /2 ...)

Les repertoires à modifier pour une application sont les suivants :

- app : vos scripts spécifiques
- dyn : le paramètrage du framework et de l application
- gen : les scripts php et sql générés
- obj : vos objets métiers en surcharge
- sql : les requetes sql surchargées
- tmp : les  fichiers temporaires
- trs : les fichiers transférés en upload.
```

Ce chapitre propose de vous décrire les outils de base du framework de la manière suivante :

- le paramétrage général du framework en /dyn
- les méthodes pour construire des formulaires avec le framework
- les outils d'édition du framework
- l'outil de requête paramétrable du framework
- la gestion des accès du framework et la multi collectivite)
- l'ergonomie intégrant jquery
- la gestion de traitement et la construction de programme spécifiques avec les utilitaires
- l'import des données CSV du framework

2.1 Paramétrage du framework

Le paramétrage de l'application se fait dans le répertoire /dyn.

Il est proposé dans ce chapitre de décrire les différents fichiers de paramétrage.

Les fichiers de paramétrage sont les suivants

```
dyn/database.inc.php           connexion a la base de données
dyn/menu.inc.php               menu principal à gauche
dyn/action.inc                 menu haut
dyn/shortslink.inc             lien sous menu haut
dyn/tbd.inc                    tableau de bord
dyn/locales.inc                application
dyn/config.inc.php             application
dyn/include.inc.php            chemin d'accès aux librairies
dyn/debug.inc.php              mode debug
dyn/version.inc                paramétrage de la version
dyn/var_sig.inc                paramétrage sig
dyn/form_sig_update.inc.php    parametrage sig
dyn/form_sig_delete.inc.php    parametrage sig

README.txt                     fichiers textes
HISTORY.txt
LICENCE.txt
TODO.txt
INSTALL.txt

app/SPECIFIC.txt                explication sur la partie spécifique de l application
```

2.1.1 La connexion de la base de donnees

Le paramétrage de la connexion se fait dans : *dyn/database.inc.php*

Le paramétrage par défaut est dans le tableau $conn[1] pour la base 1 :

Il peut être paramétré plusieurs bases : conn[1] , conn[2] . . .

conn[1] est un tableau php qui contient les parametres de connexion suivants

```
'titre          => 'openxxx',       [parametrage openmairie]
'phptype'       => 'mysql',         mysql ou 'pgsql' [parametrage dbpear]
'dbsyntax'      => '',              [ne pas changer parametrage dbpear]
'username'      => 'root',          [par defaut sur wamp easyphp ou lamp /
                                    a voir avec le fournisseur d acces le cas echeant]
'password'      => ''               [par defaut sur wamp easyphp ou lamp /
                                    a voir avec le fournisseur d acces le cas echeant]
'protocol'      => '',
'hostspec'      => 'localhost',     [nom de serveur par defaut wamp ou easyphp]
'port'          => '',              [ne pas changer parametrage dbpear]
'socket'        => '',              [ne pas changer parametrage dbpear]
'nom de la base'=> 'openxxx',       [parametrage openmairie]
'format date'   =>'AAAA-MM-JJ'      [parametrage openmairie ne pas changer]
'shema'         => ''               ou 'public' pour postgre
'prefixe'       => ''
```

Il est possible de définir tout phptype : mysql, pgsql (postgresql), oci8 pour oracle.

Il faut voir la documentation de DB PEAR qui est le module d'abstraction utilisé dans openMairie dans sa version actuelle

2.1.2 Le menu principal

Le paramétrage du menu se fait dans le fichier *dyn/menu.inc.php*.

De base, les rubriques suivantes sont paramétrées dans le framework :

```
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
                            mémorisées
traitement              vide par défaut, cet option contient les scripts de
                            traitement
parametrage             Cette option contient vos tables de paramétrage
                            et le paramètreage des états / sous états / lettretype
administration          Les scripts de cet option contiennent tout les scripts
                            du framework pour paramètrage de la collectivité,
                            om_sig  et la gestion des accès
```

Le paramétrage du menu se fait dans $menu.

$menu est le tableau associatif qui contient tout le menu de l'application, il contient lui meme un tableau par rubrique, puis chaque rubrique contient un tableau par lien :

Les caracteristiques 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)
```

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)
```

2.1.3 Le menu haut

Le paramétrage du menu haut se fait dans le fichier *dyn/action.inc.php*

Par défaut, il est paramétré le changement de mot de poste et la déconnexion

$actions est le tableau associatif qui contient tous les liens présents dans les actions à côté du login et du nom de la collectivite

les caractéristiques du tableau link sont les suivantes :

tableau link

```
title (obligatoire)
description (texte qui s'affiche au survol de l'element)
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)
```

Les liens sous le menu des actions se paramétrent dans le fichier : *dyn/shortlinks.inc.php*

$shortlinks est le tableau associatif qui contient tous les liens présents dans les raccourcis qui se situent en dessous des actions du menu haut

Par défaut, il est paramétré l'accès au tableau de bord.

Les caracteristiques du tableau $link sont les suivantes :

tableau link

```
title [obligatoire]
description (texte qui s'affiche au survol de l'element)
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)
```

2.1.4 Le tableau de bord

Le tableau de bord se paramètre dans le fichier *dyn/dashboard.inc*.

Ce fichier est appellé par le script scr/dashboard.php.

Pour avoir son propre tableau de bord, il suffit de decommenter la ligne // die() ; et on accède plus au widget

Voir chapitre : widget et tableau de bord paramétrable

2.1.5 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 ::
```

(suite sur la page suivante)

(suite de la page précédente)

```
    Dans apache, il est possible de modifiet 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'imcompatibilité utf8 de la bibliotheque fpdf␣
↪est traitée
```

— le dossier ou sont installées les variables du systeme

```
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 a traduire »)

Voir le chapitre sur les outils : *poEdit*

2.1.6 Le paramétrage de l application metier

L'application métier est paramétrée dans *dyn/var.inc*

Ce script contient les paramétres 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"]

Voir framework/utilitaire

Exemple de paramétrage avec openCourrier

```
$config['application'] = _("openCourrier");
$config['title'] = ":: "._("openMairie")." :: "._("openCourrier");
$config['session_name'] = "openCourrier";
```

— le mode demonstration de l'application se paramétre avec $config["demo"]

Ce mode permet de pre-remplir le formulaire de login avec l'identifiant "demo" et le mot de passe "demo"

```
$config['demo'] = false;  l'application n'est pas en mode démo
                  true; l'application est en mode démo

Attention, pour empêcher de changer le mot de passe, il faut paramétrer l'accès
dans la table om_droit : password
```

— La configuration des extensions autorisees dans le module upload.php

Pour changer votre configuration, décommenter la ligne et modifier les extensions avec des « ; » comme séparateur

```
$config['upload_extension'] = ".gif;.jpg;.jpeg;.png;.txt;.pdf;.csv;"
```

— Le thème de l'application

A partir de la version 3.1.0, le theme n'est plus géré dans config.inc.php. Il est initialisé dans EXTERNALS.TXT du repertoire om-theme (version 4.2.0)

```
exemple pour om_ui_darkness

om_theme svn://scm.adullact.net/svnroot/openmairie/externals/jquery-ui-theme/
                om_ui-darkness/tags/1.8.14
```

2.1.7 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

2.1.8 Le mode debug

Le mode debug d'openMairie se paramétre dans *dyn/debug.inc.php*

Ce fichier contient le paramétrage pour le mode debug d'openMairie (om_debug.inc.php)

Valeur de la variable globale DEBUG

```
EXTRA_VERBOSE_MODE : mode très bavard qui reprend les messages spécifiques
dans la méthode addToLog
exemple :
$this->addToLog("requete sig_interne maj parcelle inexistante :".$sql, VERBOSE_MODE);

VERBOSE_MODE : mode "bavard"
dans ce mode , il est créé un fielset sous les formulaires qui indiquent
toutes les étapes de réalisation des scripts
```

(suite sur la page suivante)

(suite de la page précédente)

```
DEBUG_MODE : mode debug
Les messages d'erreur sont visibles

PRODUCTION_MODE : mode de production (il n y a pas de message)
```

2.1.9 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*.

2.1.10 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 specificite 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

2.1.11 L'installation automatique

La mise en place d une installation automatique est prévue dans une prochaine version openMairie.

2.1.12 Les paramétres des combos

Les paramétres 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 :
    paramétre de l'affichage dans la fenêtre combo.php
- comboparametre.inc.php
    affecte des valeus spécifiques au formulaire parent si il y a plusieurs
    enregistrement en lien (choix en affichage)
- comboretour.inc.php
    meme chose que comboparametre.inc si il n'y a qu un enregistrement en lien
    (pas d'affichage de la fenetre)
```

Voir *chapitre framework/formulaire, sous programme générique combo.php*

2.1.13 Les paramétres éditions

Les variables dans les éditions sont paramétrées dans

```
- varpdf.inc                pour les pdf
- varetatpdf.inc            pour les états et les sous états
- varlettretypepdf.inc      pour les lettres type
```

Voir *chapitre framework/édition*

2.1.14 Les paramétres 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");
$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 paramétres sont utilisés pour la saisie de carte : voir chapitre sig

Les post traitements de form_sig permettent de faire des traitement apres saisie de géométries avec om_sig

form_sig_update.inc.php

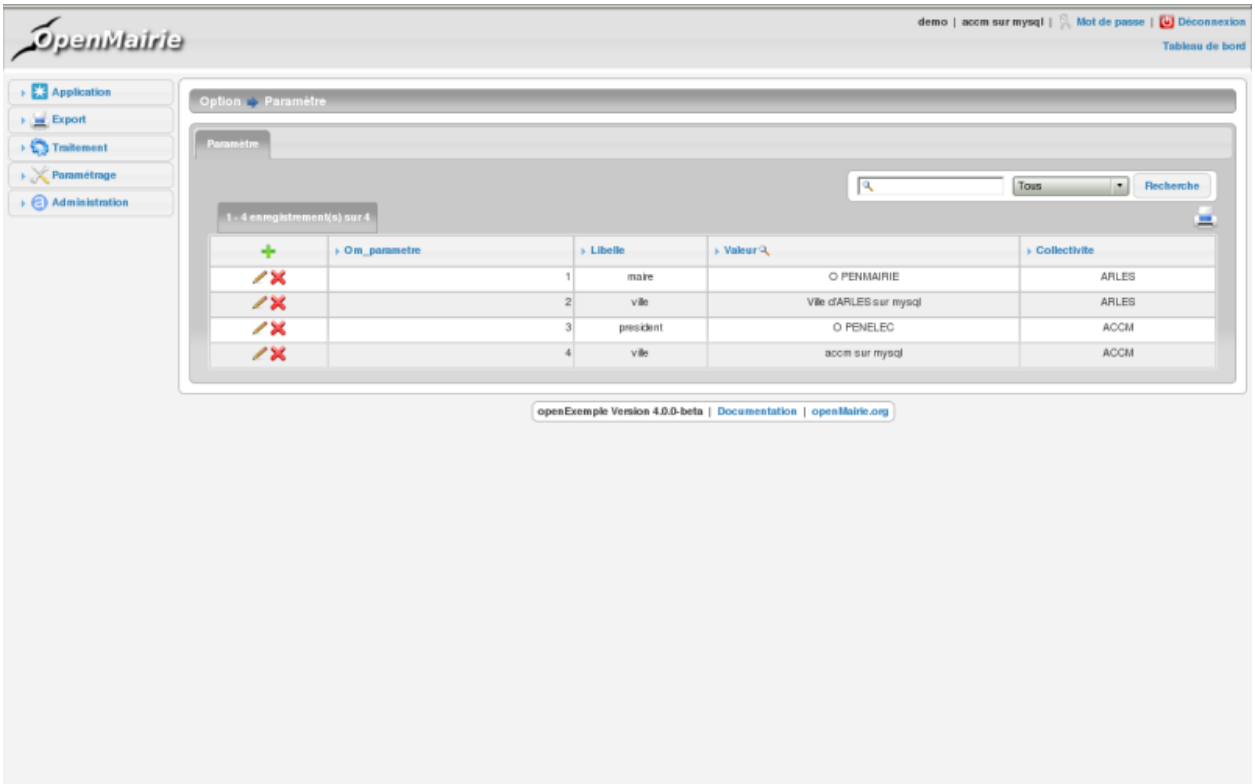
form_sig_delete.inc.php

exemple recuperation du numéro de la parcelle dans openfoncier dossier

```
if($table=="dossier" and $champ=="geom"){
   echo "</center>";
   if (file_exists ("../dyn/var.inc"))
      include ("../dyn/var.inc");
   // parcelle
   if($auto_parcelle==1){
      $sql="select parcelle from ".DB_PREFIXE."parcelle  WHERE
        ST_contains(geom,  geometryfromtext('".$geom."', ".$projection."))";
      $parcelle = $f->db -> getOne($sql);
      if($parcelle!=''){
         $sql ="update ".DB_PREFIXE."dossier set parcelle = '".$parcelle."'
            where dossier = '".$idx."'";
         $res1 =  $f->db -> query($sql);
         echo "<br>"._("parcelle")." ".$parcelle;
         // Envoi des donnees dans le formulaire f1 si la fenetre est popup
         if($popup==1){
            echo "\n<script type=\"text/javascript\">\n";
            echo "window.opener.fendata.document.f1.parcelle.value = '".$parcelle."';
↪\n";
            //echo "window.opener.fendata.reload";
            echo "</script>\n";
         }
      }
   }
....
```

2.2 Afficher les tables

Il est décrit dans ce paragraphe l'affichage de requete sous forme de table pour faire un choix d'ajout, de mise à jour ou de suppression.



2.2.1 La requete SQL d'affichage

Elle se trouve dans sql/type_de_sgbd/nom_objet.inc

Les paramétres sont les suivants pour om_parametre.inc

```
$serie=15;                                          Nombre d'enregistrement par page

$ico="../img/ico_application.png";                  Icone affiché (a voir deprecated)

$ent = _("option")." -> "._("om_parametre");        Titre du tableau

$idz                                                affichage en haut du formulaire

$table=DB_PREFIXE."om_parametre";                   Table de référence
                                                    (il peut y avoir une ou plusieurs␣
↪jointure)
$champAffiche=array('om_parametre',
                    'libelle',
                    'valeur',
                    'om_collectivite');

$champRecherche=array('libelle','valeur');          Champs pour la recherche
```

(suite sur la page suivante)

(suite de la page précédente)

```
$tri="";                                        Critere de tri par défaut

$edition="om_parametre";                        edition pdf

$sousformulaire= array()                        sous formulaire(s) associé(s)



autre exemple de sous formulaire avec om_collectivite.inc

    $sousformulaire=array('om_etat',
                    'om_lettretype',
                    'om_parametre',
                    'om_sousetat',
                    'om_utilisateur');
```

2.2.2 Le script scr/tab.php

L'affichage se fait à partir du menu (voir *framework/parametrage*) sous la forme

```
tab.php?obj=om_parametre

où obj = nom_d_objet
```

2.2.3 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

2.3 Les formulaires

Les formulaires se construisent sur la base de la classe om_formulairedyn.class.php d'openMairie

Cette classe fait appel a des sous programmes generiques pour certains controles au travers de script js/script.js

2.3.1 Les methodes de om_formulairedyn.class.php

La classe om_formulaire.class.php a les méthodes suivantes :

Les méthodes sur les controles du formulaire

```
text : Controle text (format standart)
hidden : Controle non visible avec valeur conservée
password : Controle password
textdisabled : Controle text non modifiable
textreadonly : contrôle text non modifiable
hiddenstatic : Champ non modifiable  Valeur récupéré par le formulaire
```

(suite sur la page suivante)

(suite de la page précédente)

```
hiddenstaticnum : champ numerique non modifiable et valeur récupérer
statiq : Valeur affichée et non modifiable
affichepdf : récupére un nom d'objet (un scan pdf)

checkbox : controle case à cocher : cochée = Oui, Non cochée = Non
checkboxnum : cochée = 1 , non cochée = 0

http : lien http avec target = _blank (affichage dans une autre fenêtre)
httpclick : lien avec affichage dans la même fenêtre.

date et date2 : date modifiable  avec affichage de calendrier jquery
Hiddenstaticdate  date non modifiable Valeur récupéré par le formulaire

textarea : affichage d un textarea
textareamulti : textarea qui récupére plusieurs valeurs d'un select
textareahiddenstatic : affichage non modifiable d'un textarea et recupération de la␣
↪valeur
pagehtml : affichage d'un textarea et tranforme les retour charriot en <br>

select : Controle select
selectdisabled : Controle select non modifiable

comboG  et comboG2-> Appel à un programme de correspondance à une table
           Cas ou il y a une grosse table en correspondance
           spg/combo.php
ComboD et comboD2 -> Appel à un programme de correspondance à une table
           Cas ou il y a une grosse table en correspondance
           spg/combo.php

Upload et upload2 fait appel à spg/upload.php pout télécharger un fichier
voir et voir2 : fait appel à spg/voir.php pour visualiser un fichier

localisation et localisation2 : fait appel à spg/localisation.php
rvb et rvb2 : fait appel à spg/rvb.php pour affichage de la palette couleur

geom : ouvre une fenetre tab_sig.php pour visualiser ou saisir une geometrie (si maj)
       la carte est définie en setSelect

Les contrôle comboG, comboD, date, upload, voir et localisation sont à mettre dans
les formulaires (retour de l'affichage dans le formulaire f1)
Les contrôle comboG2, comboD2, date2, upload2, voir2 et localisation sont à mettre␣
↪dans
les sous formulaires (retour de l'affichage dans le formulaire f2)
```

Les méthodes de construction et d affichage

```
afficher() affichage des champs (appelle par om_dbformdyn.class.php : methode␣
↪formulaire
        -> afficherChampRegroupe() affichage des champs par regroupement / groupement
        -> afficherChamp() affichage de champ sans regroupe
recupererPostvarsousform() et recuperePostVar():
               recupèrent des variables apres validation
enpied() presentation
```

Les méthodes assesseurs changent les valeurs des proprietes de l'objet form (formulaire)

```
setType()
setVal()
setLib()
setSelect()
setTaille()
setMax()
setOnchange()
setKeyup()
setOnclick()
setSelect()
setGroupe()
    D premier champ du groupe
    G champ groupe
    F dernier champ du groupe
setRegroupe()
    D premier champ du fieldset
    G champ dans le fieldset
    F dernier champ du fieldset
```

et enfin les méthodes de date

```
dateAff($val)
```

2.3.1.1 Les sous programmes génériques

Les sous programmes génériques sont des sous programmes associés aux contrôles du formulaire et appellés par eux par un script js dans js/formulairedyn.js

Les sous programmes génériques sont stockés dans le répertoire /spg.

spg/combo.php

Ce programme est appellé par le contrôle 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 contrôle formulaire « localisation »

spg/voir.php

Ce script est associé au contrôle « 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 le fichier autorise dans dyn/config.inc.php

spg/rvb.php et js/rvb.js

Ce script est associé au contrôle « rvb » et permet l'accès à une palette de couleur pour récupérer un code couleur rvb

2.3.1.2 le script scr/form.php

form.php est le programme appellant d'un formulaire par rapport à un objet métier(om_parametre) et un identifiant (2)

form.php affiche le formulaires et éventuellement les sous formulaires (soustab.php et sousform.php)

exemple

```
form.php?obj=om_parametre&idx=2
```

Les méthodes de core/om_formulaire.class.php peuvent être surchargées dans obj/om_formulaire.class.php

Les scripts javascript de js/script.js peuvent être surchargés dans app/js/script.js

Les méthodes de core/om_dbform.class.php peuvent être surchargées dans obj/om_dbform.class.php

2.3.1.3 Les nouvelles utilisations dans les objets metiers (openMairie 4)

openMairie4 apporte de nouvelles fonctions qu'il est utile d'implémenter dans les objets métiers

récuperer le type de la base depuis l'objet db : $db->phptype (mysql ou pgsql) :

```
if(file_exists ("../sql/".$db->phptype."/".$this->table.".form.inc"))/
                /include ("../sql/".$db->phptype."/".$this->table.".form.inc");/
```

récuperer une erreur dans la base

om4

```
database::isError($res); // ($res,true) = sans die
```

ce code remplace le code om3 (deprecated)

```
//   if (DB :: isError($res))
//            $this->erreur_db($res->getDebugInfo(),$res->getMessage(),'');
//    else
//    {
//    if ($DEBUG == 1)
//            echo "La requ&ecirc;te de mise &agrave; jour est effectu&eacute;e.<br>";
```

2.4 La méthode

Il est décrit ici la méthode pour la création d" objets métiers :

Le développement consiste à créer des objets métier (/obj) qui surchargent la classe abstraite om_dbformdyn.class.php et à modifier les valeurs par défaut des variables dans les fichiers sql (nom_objet.inc et nom_objet.form.inc)

Voir aussi le *générateur* pour automatiser les scripts métier.

2.4.1 Surcharger les classes openMairie

Il vaut mieux utiliser le générateur pour initialiser les classes metiers.

Le générateur surcharge la classe om_dbformdyn.class.php par rapport aux informations de la base

```
classe abstraite <- classe metier generee <- classe metier 1 <- classe metier 2 ...
openMairie             depuis la base


om_dbformdyn.class.php <- gen/obj/nom_objet.class.php <- obj/nom_objet.class.php
```

Exemple avec concession d'openCimetiere

```
om_dbformdyn.class.php
        <- gen/obj/emplacement.class.php
            <-/obj/emplacement.class.php <- /obj/concession.class.php
```

2.4.2 Modifier les valeurs par défaut

Il est décrit ici les valeurs par défaut dans core/om_dbformdyn.class.php qui est une classe d'openMairie.

Les valeurs suivantes sont mises par defaut afin de pouvoir construire rapidemment un formulaire

```
valeur par defaut
   en ajout = initialisation vide

type par defaut
   type text pour ajout et modification
   type hiddenstatic pour suppression

libelle par défaut :
   Libellé = nom du champ dans le SGBD

taille et max d un champ
   Taille et max = longueur du champ dans le SGBD

les regroupements et groupements de champs sont vides

les fonctions javascript ne sont pas utilisées
```

2.4.3 Modifier les valeurs par defaut par les méthodes assesseurs

Elles se font dans la classe obj/nom_objet.class.php

Les valeurs par défaut sont modifiées par la méthode setVal(nomduchamp, nouvelle valeur)

Les types par défaut sont modifiés par la méthode setType(nomduchamp, nouveau type)

Les longueurs d affichage par défaut sont modifiées par la méthode setTaille(nomduchamp, nouvelle valeur)

Les maximums autorisés par défaut sont modifiés par la méthode setMax(nomduchamp, nouvelle valeur)

Les libelles de champ par défaut sont modifiés par la méthode setLib(nomduchamp, nouvelle valeur)

Les scripts javascript sont appellés dans la méthode setOnchange()

Voir framework/formulaire

2.4.4 La class om_dbformdyn.class.php

om_dbform.class.php est une classe openMairie dans core/

La classe abstraite dbform gère l'interface entre l'objet métier et la base de données connectée via DBPEAR.

Les méthodes principales sont les suivantes :

— orientées sgbd

```
constructeur
ajouter : Ajoute un objet
Modifier : Modifie un objet
Supprimer : Supprime un objet
Verifier : Contrôle un objet
Clesecondaire : Contrôle les cles secondaires
triggers avant/apres ajout/modification/suppression
```

— orientees Formulaire

```
Formulaire : Constitue le formulaire et fait appel à formulaire.dyn.class.php
sousFormulaire : Constitue le sousformulaire -> appel à formulaire.dyn.class.php
Message : Retourne le message d erreur (contrôle php)
bouton : Affiche le bouton
Retour : gére le retour à une interface php en fin de saisie
sousformulaireRetour : gére le retour à une interface php en fin de saisie de␣
↪sous formulaire
setType : Envoi au formulaire les type de champ
setVal : Envoi au formulaire les valeurs par défaut
setValSousformulaire : Envoi au sousformulaire les valeurs par défaut
setlib : Envoi au formulaire les libellés de champs
setTaille : Envoi au formulaire la taille du champ
setMax : Envoi au formulaire la taille maximum autorisée du champ
setSelect : Envoi au formulaire les champs select à afficher
setOnchange : Envoi au formulaire les controles javascript à effectuer en cas de␣
↪changement de données dans le champ
setGroupe : Envoi au formulaire le groupement de champ par ligne
setRegroupe : Envoi au formulaire un fieldset
setOnkeyup
setOnclick
mail
selectiste
selectlistemulti
```

— des fonctions de traitement de champ heure et date :

```
DateDB : transforme les dates affichées en date pour base de données
HeureDB : controle du champs heure saisi 00 ou 00:00 ou 00:00:00
DateSystemeDB : mise au format base de donnees de la date systeme
DatePHP : controle et transforme la date saisie (jj/mm/aaaa) en date format PHP
```

— des fonctions pour faire des calculs

```
AnneePHP : controle et recupere l'année de la date saisie (jj/mm/aaaa)
MoisPHP : controle et recupere le mois de la date saisie (jj/mm/aaaa)
JourPHP : controle et recupere le jour de la date saisie (jj/mm/aaaa)
```

La classe dbformdyn.class.php fait appel à la classe formulaire.dyn.class.php pour afficher le formulaire.

Il est créé 2 objets :

— un objet db qui fait la connexion avec la base
— un objet form qui décrit le formulaire

2.4.5 L'objet db

db est l'objet de connexion a la base dont les proprietes sont les suivantes

```
DB_pgsql Object

(
[phptype] => pgsql
    [dbsyntax] => pgsql
    [features] => Array (
                    [limit] => alter
                    [new_link] => 4.3.0
                    [numrows] => 1
                    [pconnect] => 1
                    [prepare] =>
                    [ssl] => 1
                    [transactions] => 1 )
                    [errorcode_map] => Array ( )
                    [connection] => Resource id #19
                    [dsn] => Array (
                            [phptype] => pgsql
                            [dbsyntax] => pgsql
                            [username] => postgres
                            [password] => postgres
                            [protocol] => tcp
                            [hostspec] => localhost
                            [port] => 5432
                            [socket] =>
                            [database] => sig
                            [title] => Openmairie Exemple PostGreSQL schema SIG
                            [formatdate] => AAAA-MM-JJ
                            [schema] => openmairie
                    )
                    [autocommit] => 1
                    [transaction_opcount] => 0
                    [affected] => 0
                    [row] => Array ([20] => 10 )
                    [_num_rows] => Array ( [20] => 10 )
                    [fetchmode] => 1
                    [fetchmode_object_class] => stdClass
                    [was_connected] =>
                    [last_query] => select * from openmairie.om_parametre where om_
↪collectivite=2
                    [options] => Array (
            [result_buffering] => 500
                            [persistent] =>
                            [ssl] =>
            [debug] => 2
            [seqname_format] => %s_seq
            [autofree] =>
            [portability] => 63
            [optimize] => performance
            )
                    [last_parameters] => Array ( )
                    [prepare_tokens] => Array ( )
                    [prepare_types] => Array ( )
                    [prepared_queries] => Array ( )
                    [_last_query_manip] =>
```

(suite sur la page suivante)

(suite de la page précédente)

```
                    [_next_query_manip] =>
                    [_debug] =>
                    [_default_error_mode] =>
                    [_default_error_options] =>
                    [_default_error_handler] =>
                    [_error_class] => DB_Error
                    [_expected_errors] => Array ( )
)
```

2.4.6 L'objet form

form est l'objet formulaire dont les proprietes sont les suivantes

```
formulaire Object (
    [enteteTab] =>
    [val] => Array (
            [om_parametre] => 1
            [libelle] => maire
            [valeur] => O PENMAIRIE
            [om_collectivite] => 1 )
    [type] => Array (
            [om_parametre] => text
            [libelle] => text
            [valeur] => text
            [om_collectivite] => text )
    [taille] => Array (
            [om_parametre] => 11
            [libelle] => 20
            [valeur] => 50
            [om_collectivite] => 11 )
    [max] => Array (
            [om_parametre] => 11
            [libelle] => 20
            [valeur] => 50
            [om_collectivite] => 11 )
    [lib] => Array (
            [om_parametre] => Om_parametre
            [libelle] => Libelle
            [valeur] => Valeur
            [om_collectivite] => Om_collectivite )
    [groupe] => Array (
            [om_parametre] =>
            [libelle] =>
            [valeur] =>
            [om_collectivite] => )
    [select] => Array (
            [om_parametre] =>  Array ([0] => [1] => )
            [libelle] => Array ( [0] => [1] => )
            [valeur] => Array ( [0] => [1] => )
            [om_collectivite] => Array ( [0] => [1] => ) )
    [onchange] => Array (
            [om_parametre] =>
            [libelle] =>
            [valeur] =>
            [om_collectivite] => )
```

(suite sur la page suivante)

(suite de la page précédente)

```
    [onkeyup] => Array (
            [om_parametre] =>
            [libelle] =>
            [valeur] =>
            [om_collectivite] => )
    [onclick] => Array (
            [om_parametre] =>
            [libelle] =>
            [valeur] =>
            [om_collectivite] => )
    [regroupe] =>
    [correct] =>
)
```

2.5 Les éditions

Les éditions sont accessibles dans le menu par

```
- administration -> etat
- administration -> sousetat
- administration -> lettretype
```

Depuis la version 4 d'openMairie, les editions sont conservées dans 3 tables

```
- om_etat : pour les états
- om_sousetat : pour les sous etats
- om_lettretype : pour les lettres types
```

Cette modification a été faite pour pouvoir gérer la multi collectivité.

Par contre, les tableaux pdf sont stockés dans un fichier : nom_objet.pdf.inc

2.5.1 Actif, non actif

Les sous etats sont liés a un ou plusieurs état

Les états, sous etats, et lettre type peuvent être actif ou non actif

Par défaut sont pris en compte :

1 - l'édition « actif » de la collectivite

2 - l'édition « actif » de la multicollectivite

3 - l'édition « non actif » de la multicollectivite

Les editions non actifs d'une collectivite ne sont pas pris en compte

2.5.2 Paramétrer des etats

Il est conseillé d utiliser l'assistant état du generateur

Les paramètres sont les suivants

```
orientation portrait ou paysage
format="A4", A3
position et nom  du logo
titre de l etat
position et caractéristiques du titre
corps de l etat
position et caractéristiques du corps
la requete SQL
les sous etats associés et les caractéristiques
```

Pour le corps et le titre, les zones entre crochets (exemple [nom]) sont les champs selectionnés par la requete.

Les variables commençant par « & » sont définies dans dyn/varpdf.inc (exemple &aujourdhui) et dans la table om_parametre.

2.5.3 Paramétrer des sous etats

Il est conseillé d utiliser l'assistant sousetat du générateur

Les paramétres sont les suivants

```
texte et caractéristique du Titre
Intervalle avant et apres le tableau
Entete de tableau (nom de colone)
caracteristique du tableau
caracteristique des cellules
tatal, moyenne, nombre
requete sql
```

Pour le titre, les zones entre crochets sont les champs selectionnés par la requete.

Les variables commençant par « & » sont définies dans dyn/varpdf.inc (exemple &aujourdhui) et dans la table om_parametre

2.5.4 Paramétrer des lettres type

Il est conseillé d utiliser l assistant lettretype du generateur

Les paramétres 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 requete SQL
```

Pour le corps et le titre, les zones entre crochets sont les champs selectionnés par la requete.

Les variables commençant par « & » sont définies dans dyn/varlettretypepdf.inc (exemple &aujourdhui) et dans la table om_parametre

2.5.5 Parametrer des edition pdf

Un etat pdf peut être généré par le generateur (option)

L'edition est paramétrée dans un fichier sql/sgbd/nom_objet.pdf.inc et dans la

Les paramétres sont les suivants

```
texte et caractéristique du Titre
Entete de tableau (nom de colone)
caracteristique du tableau
caracteristique des cellules
tatal, moyenne, nombre
requete sql
```

Pour le titre, les zones entre crochets sont les champs selectionnés par la requete.

Les variables commençant par « & » sont définies dans dyn/varpdf.inc (exemple &aujourdhui) et dans la table om_parametre

2.5.6 Parametrer les etiquettes

Les zones entre crochets sont les champs selectionnés par la requete. La variable &aujourdhui sont définies dans dyn/varetiquettepdf.inc et dans la table om_parametre

Il y aura une integration depuis l utilisation d'openPersonnalite dans une prochaine version openMairie.

2.5.7 L'éditeur WYSIWYG

Un editeur est prevu dans une prochaine version openMairie.

2.5.8 Les scripts PDF

Les scripts sont dans le répertoire **pdf/** et sont appellés par le framework sous la forme

```
pdfetat.php?obj=nom_etat&idx=enregistrement_a_editer
```

les scripts sont les suivants

```
pdfetat.php : etat et sous etat
pdf.php : edition pdf
pdfetiquette.php : etiquette
pdflettretype.php
```

pdfEtiquette sera repris dans une prochaine version d'openMairie

specifique openCourrier pour ecriture sur pdf

```
fpdf_tpl.php
fpdi.php
fpdi2tcpdf_bridge.php
fpdi_pdf_parser.php
histo.htm
pdf_context.php
pdf_parser.php
testfpdi.php
```

Il n est pas prévu d integration dans la prochaine version

2.5.9 composants

/core

Les scripts ci dessous sont les classes qui interfacent openmairie avec fpdf

```
fpdf_etat.php
fpdf_etiquette.php
db_fpdf.php
```

php/fpdf

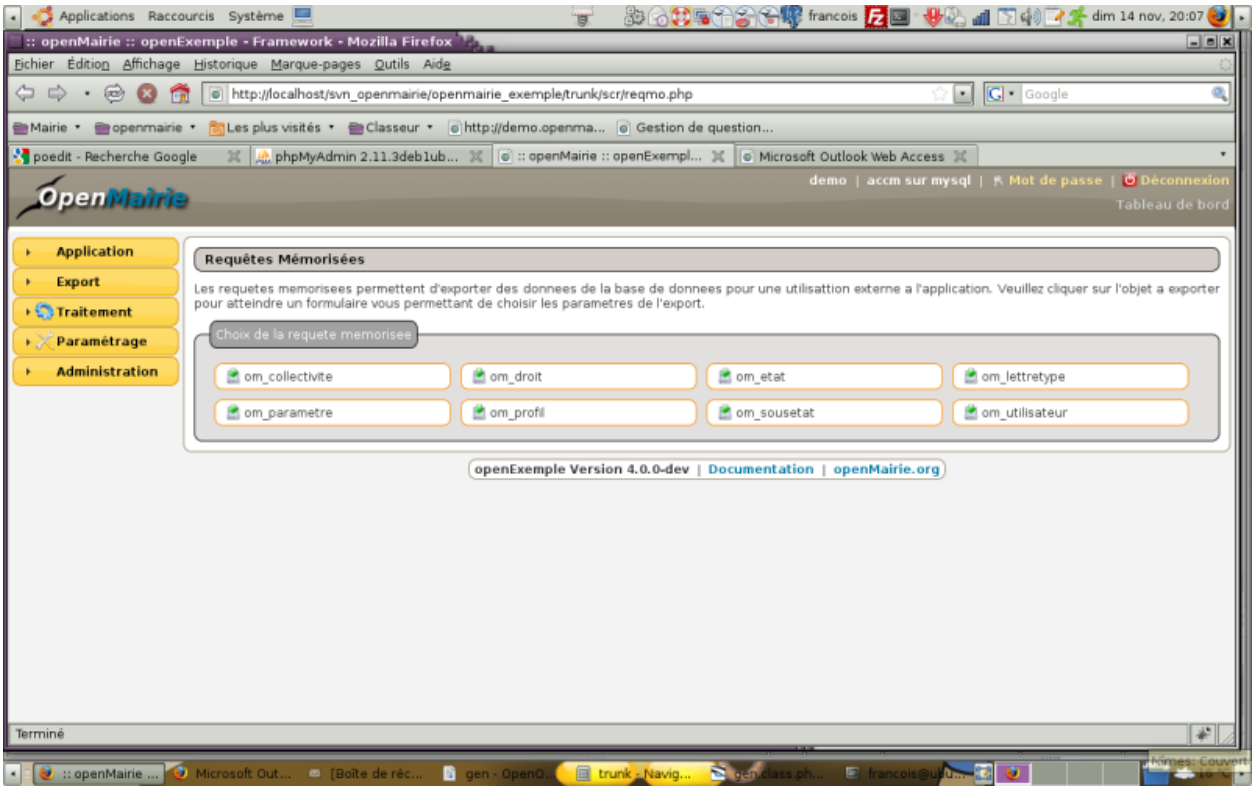
A ce niveau se situe le composant fpdf

2.6 Les requêtes memorisées

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



2.6.1 Description du parametrage

Les parametres de reqmo sont :

$reqmo[“libelle”] contient le libéllé affiché en haut
$reqmo[“sql”] contient la requete SQL.
Dans la requete, les paramétres sont mis entre []
et ils sont définis en dessous sous la forme reqmo[parametre]=.

« checked » : la colonne est affiché ou non
« un tableau » array(a,b) et le choix a ou b est donné à l utilisateur de requete
« une requete sql » : le choix se fait dans la table du select

La requete executée est celle qui est reconstituée avec les zones sasisies par l'utilisateur

Enfin, l'utilisateur choisit soit un affichage soit en tableau, soit en csv avec un choix de séparateur.

Il n y a pas d'outil de fabrication de requête à part l'option du générateur (voir chapitre sur le *générateur*)

2.6.2 Exemple

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";
```

2.7 La gestion des accès

Le framework fournit un gestionnaire d'accés accessible dans le menu à

```
- administration -> profil
- administration -> droit
- administration ->utilisateur
```

Les accès sont conservés dans des tables.

2.7.1 Les tables

La gestion des accès est gérée avec 3 tables :

om_profil : gestion des profils

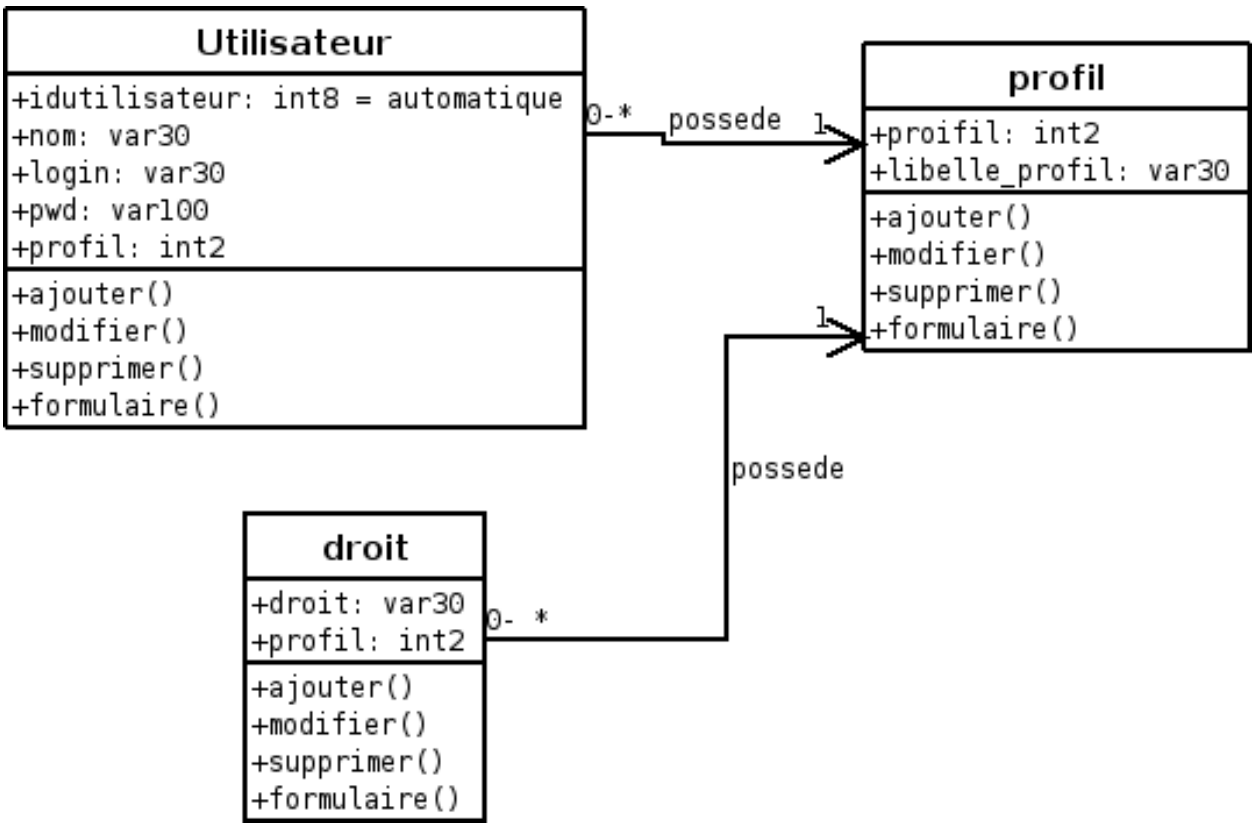
```
administrateur
super utilisateur
utilisateur
utilisateur limite
consultation
```

om_droit : la gestion des droits affecte un profil suivant chaque :

objet métier : $obj om_collectivite, om_parametre . . .
chaque rubrique du menu :
voir paramétrage menu : tableau Rubrik right = « om_parametre »

om_utilisateur : cette table permet de donner un login, un mot de passe et un profil à chaque utilisateur

Diagramme de classe



2.7.2 Les règles

— le droit sur un objet porte le nom de l'objet, avec l'extension _tab, il porte sur l'affichage en table de l'objet exemple om_droit d'om_utilisateur :

```
om_utilisateur = 5      en form.php?obj=om_utilisateur :
                 accès en maj permis au utilisateurs de niveau 5 et plus
om_utilisateur_tab = 4  en tab_php?obj=om_utilisateur :
                 acces en lecture de table qu aux utilisateurs de niveau 4 et plus
```

— accès à la rubrique se paramètre dans le menu et dans om_droit

exemple

```
menu_administration = 3
        cette rubrique n'apparait qu'aux utilisateurs d'om_profil  supérieur ou égal␣
↪à 3
```

— chaque profil a acces a tous les droits des profils d un niveau inférieur
— l'administrateur a acces à tout.

2.7.3 Les login et logout

Le login se fait par le script *scr/login.php*

login.php valorise les variables sessions permettant la gestion des acces et securites :

```
$_SESSION['profil'] = $profil;
$_SESSION['nom'] = $nom;
$_SESSION['login'] = $login;
```

La deconnexion se fait avec le script *scr/logout*

Le changement de mot de passe se fait avec le script *scr/password.php*

L'accès au changement de passe se fait par défaut dans le menu haut (voir framework/paramétrage)

2.7.4 Les utilitaires

La gestion des droits d'acces se fait dans les méthodes des utilitaires

php/openmairie/om_appication.class.php (composant openMairie)
obj/utils.class.php

(*voir framework/utilitaire*)

2.8 L'ergonomie

Depuis la version openMairie 4, il est utilisé l'ergonomie de jquery.

2.8.1 Le composant jquery

Les skins jquery peuvent être rajoutés dans le repertoire /om_theme/.

Le changement de skin peut se faire dans le fichier EXTERNALS.txt

voir framework/parametrage

2.8.2 Les feuilles de style

Les feuilles de style sont stockées dans le repertoire css/ et sont cascadables

```
main.css : 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)
```

2.9 Les scripts specifiques de l'application

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 openCourrier.

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 appellé 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

2.9.1 Réaliser un script complementaire

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 paramétre « right » est vide vous pouvez faire appel aux méthodes suivantes

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

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

// 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 utilisez 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()
```

2.9.2 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 $
 */
```

(suite sur la page suivante)

(suite de la page précédente)

```
// 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");



// 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 ($res1!=0){
        echo "<br>"._("le dernier no du registre est")." :   ".$res1." 
↪ ";
    }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\">";
```

(suite sur la page suivante)

(suite de la page précédente)

```
    echo "<input type='submit' value='"._("remise a 0 du registre").
          " ' >";
    echo "</div>";
    echo "</form>";

}else { // validation=1

    // VALORISATION DE $valF
    $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

2.10 Importer des données en csv

Il est possible d'importer des données suivant des scripts pré paramétrées mais qui sont modifiables.

Pour lancer le menu import, prenez l'option : administration -> import

import_script.php permet les imports dans la base de donnees de fichier au format csv telecharge :

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"
```

La description du transfert se fait dans le fichier extension import_nomobjet.inc dans /sql/. . . :

exemple : import_script.php ?obj=utilisateur

dans utilisateur.import .inc , il est defini :

```
le message affiché en import :
    $import= "Insert utilisateur" : Message

la table d importation
    $table= "utilisateur"
```

(suite sur la page suivante)

(suite de la page précédente)

```
le 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 sonnées
    $verrou = 1  mise a jour de la base
            = 0 pas de mise a jour pour une phase de test

Le mode debug
    $DEBUG  =1 affichage des enregistrements a l ecran
            =0 pas d affichage

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 apres 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 premiere 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

    $exist['profil']=1 => 0=non / 1=oui
    $sql_exist["profil"]= "select 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 2éme zone contient le login
    $zone['pwd']='2' => la 3éme zone contient le mot de passe (crypte)
    $zone['profil']='3' => la 4éme zone contient le profil

La valeur par defaut :
En effet, si $zone['profil']="" on peut definir un profil par defaut
    $defaut['profil']='5' Le profil par defaut  sera 5
```

CHAPITRE 3

Le générateur

Nous vous proposons dans ce chapitre de décrire le générateur openMairie.

- présentation du générateur
- les écrans du générateur
- l'analyse de la base
- les fichiers générés
- le paramétrage du générateur
- les vues (avec postgresql)

3.1 Présentation

L'objectif est de construire une application sur la base de l'analyse des informations du SGBD

Les informations récupérées dans le SGBD sont les suivantes

```
la liste des tables de la base de données

les tables : nom, type , et longueur de chaque champs
```

Le générateur construit sur cette base le modèle de données sur les principes suivants :

```
le nom de la clé primaire est le nom de la table et c'est le premier champ

la clé secondaire est le nom de la table en lien

si la clé est numérique, elle est automatique.

avec la multicollectivité, la création d'un champ « om_collectivite »
met en place les accès multicollectivités d'openMairie 4
```

Les assistants vont faciliter la mise en oeuvre des états

Il est fourni avec le générateur un assistant pour faire les états et les sous états.

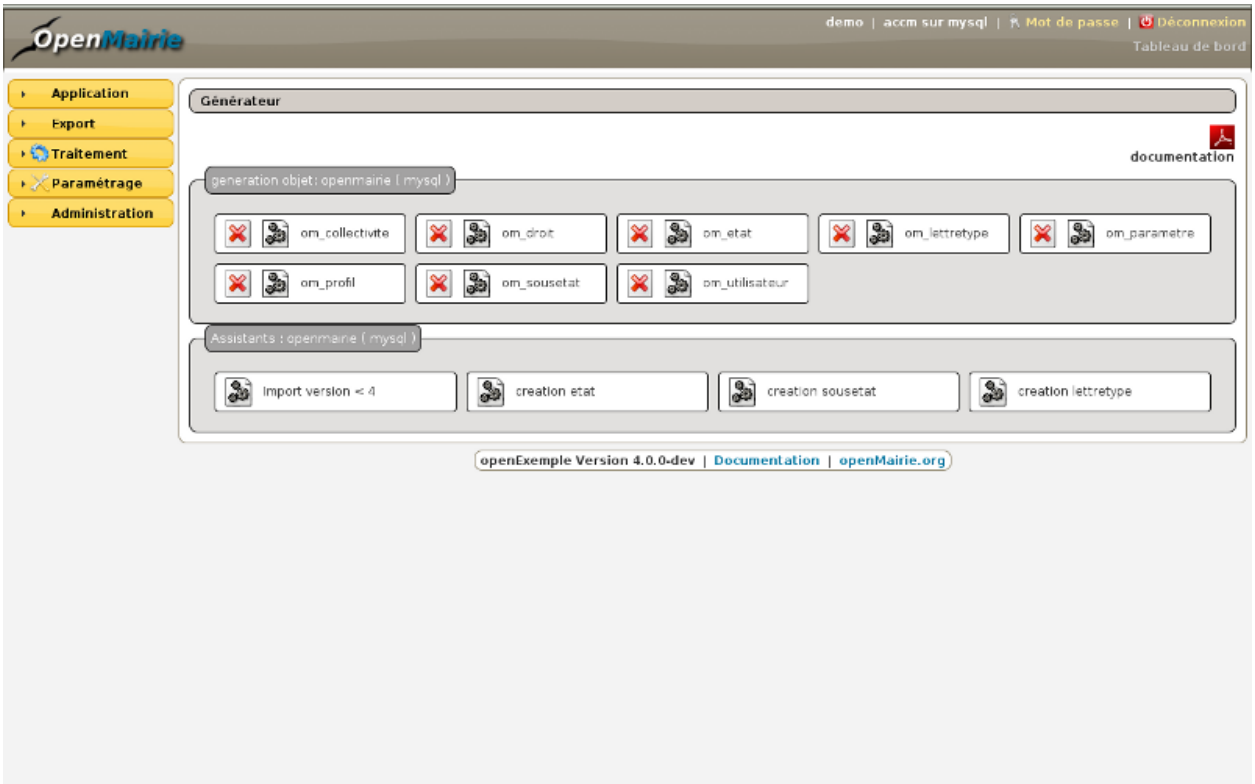
openMairie est multi collectivité : les états et les sous états sont générés dans la base de données et peuvent être associé a une collectivité.

Le générateur gére la multicollectivité si un champ « om_collectivité » est créé.

Les schemas et prefixes sont gérés

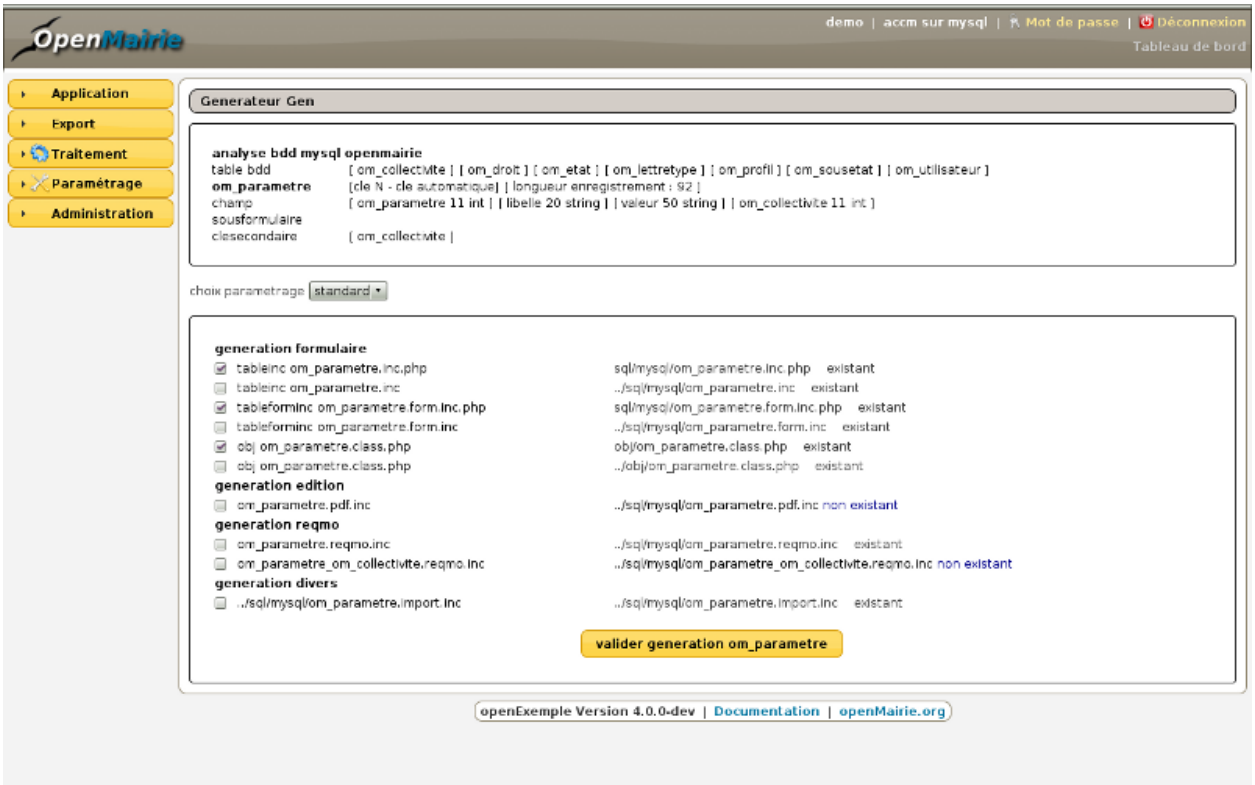
3.2 Les écrans du générateur

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



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_colllectivite)
- 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.

3.2.2 Les fichiers à generer

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 = mysql ou pgsql

3.3 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 ou décimal)
- date
- blob : texte
- geom : geometry (pour postgres)
```

3.3.1 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)

3.3.2 Equivalence type mysql / type openMairie

type mysql (longueur) tableinfo -> type openMairie

```
Int          (taille mysql)                  -> Int
Date         (10)                            -> date
Blob         (65535)                         -> Blob
Char         (taille mysql)      char        -> String
tinyint      (4)                 tinyint     -> Int
smallint     (6)                 smallint    -> Int
Mediumint    (9)                 mediumint   -> Int
Bigint       (20)                bigint      ->Int
Float        (12)                Real        -> Int
Double       (22)                Real        -> Int
Decimal      (11)                Real        -> Int
```

(suite sur la page suivante)

(suite de la page précédente)

```
Text         (65535)              Blob        -> Blob
Tinyblob     (255)                blob        ->Blob
Mediumblob   16777615             Blob        -> Blob
Mediumtext   16777215             Blob        -> Blob
Longtext     -1                   blob        -> Blob
Longblob     -1                   Blob        -> Blob
Tinytext     255                  Blob -      -> blob
```

3.3.3 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 different -> type openMairie

```
Bigint       (8)                  int8        -> int
Smallint     (2)                  Int2        -> Int
Integer      (4)                  Int4        -> Int
Real         (4)                  Float4      -> Int
Doubleprecision (8)               Float8      -> Int
Numeric      (20)                 Numeric     -> Int
Money        (8)                  Money       -> Int
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
```

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_defaut = 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);
}
```

3.3.4 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

Les règles suivantes sont spécifiques au générateur pour reconnaître les clés primaires et les clés secondaires :

la cle primaire de la table a le même nom que la table

le cle secondaire a le même nom que la table fille

3.4 Les fichiers générés

Les fichiers générés concernent :
- les formulaires
- les requêtes mémorisées
- le script d'import de données

3.4.1 Les formulaires

Les formulaires sont génèrés 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.

3.4.1.1 Paramétres 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 defaut

3.4.1.2 Paramétres 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 defaut

Par défaut : - tous les champs sont affichés les uns en dessous des autres

3.4.2 Les 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 regénérer tout ou partie de l'application

Un second script héritant de l'objet généré permet de surcharger les méthodes et de personnalisé 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
- 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
```

(suite sur la page suivante)

(suite de la page précédente)

```
om_utilisateur.class.php :
    champ pwd pour mot de passe  methode partiellement surchargées (parent::setvalF(
↪$val);)
    setvalF, setType, setValsousformulare,
    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)
- 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

3.4.3 Les états

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'etat 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.

3.4.4 les requêtes mémorisées

Les requêtes paramétrées sont crées 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.

3.4.5 les 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.

3.5 Paramétrage générateur

Le paramétrage de base est dans la classe gen Il est possible de personnaliser le paramétrage dans le répertoire gen/dyn. (version 4.2.0)

Ne personnaliser que les variables souhaitées dans le fichier. Par défaut, openMairie prendra les paramètres inclus dans la classe gen.

Il est donné ci dessous l ensemble des paramètres « customizable ». La valeur associée est celle du générateur.

3.5.1 Form.inc

Voici les paramètres pour la génération de formulaire

```
$serie = 15;                          nombre d'enregistrement par page'
$ico ="../img/ico_application.png"; icone DEPRECATED
$max=6;                               nb de ligne blob
$taille=80;                           taille du blob
$pgsql_longueur_date=12;              taille d'affichage de la date '

*** deprecated
$pgsql_taille_defaut = 20;            taille du champ par defaut si retour pg_field_
↪prtlen =0
$pgsql_taille_minimum    = 10;        taille minimum d affichage d un champ
***/
```

3.5.2 pdf.inc.php

Parametres

```
$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";
```

(suite sur la page suivante)

(suite de la page précédente)

```
$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";
$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";
$heightitre=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";
$C1enteteetxt=0;// couleur texte R";
$C2enteteetxt=0;// couleur texte V";
$C3enteteetxt=0;// couleur texte B";
$C1border=159;// couleur texte  R";
$C2border=160;// couleur texte  V";
$C3border=167;// couleur texte  B";
$bt=1;// border 1ere  et derniere ligne  du tableau par page->0 ou 1";
```

3.5.3 etat.inc.php

parametres

```
$variable='&'; // nouveau
// parametres
$etat['orientation']='P';
$etat['format']='A4';
// footer
$etat['footerfont']='helvetica';
$etat['footerattribut']='I';
$etat['footertaille']='8';
// logo
$etat['logo']='logopdf.png';
$etat['logoleft']='58';
```

(suite sur la page suivante)

(suite de la page précédente)

```
$etat['logotop']='7';
// titre
$etat['titreleft']='41';
$etat['titretop']='36';
$etat['titrelargeur']='130';
$etat['titrehauteur']='10';
$etat['titrefont']='helvetica';
$etat['titreattribut']='B';
$etat['titretaille']='15';
$etat['titrebordure']='0';
$etat['titrealign']='C';
// corps
$etat['corpsleft']='7';
$etat['corpstop']='57';
$etat['corpslargeur']='195';
$etat['corpshauteur']='5';
$etat['corpsfont']='helvetica';
$etat['corpsattribut']='';
$etat['corpstaille']='10';
$etat['corpsbordure']='0';
$etat['corpsalign']='J';
// sous etat
$etat['se_font']='helvetica';
$etat['se_margeleft']='8';
$etat['se_margetop']='5';
$etat['se_margeright']='5';
$etat['se_couleurtexte']="0-0-0";
```

3.5.4 sousetat.inc.php

parametres :

```
$longueurtableau= 195;
$variable='&'; // nouveau
// parametres

//titre
$sousetat['titrehauteur']=10;
$sousetat['titrefont']='helvetica';
$sousetat['titreattribut']='B';
$sousetat['titretaille']=10;
$sousetat['titrebordure']=0;
$sousetat['titrealign']='L';
$sousetat['titrefond']=0;
$sousetat['titrefondcouleur']="255-255-255";
$sousetat['titretextecouleur']="0-0-0";
// intervalle
$sousetat['intervalle_debut']=0;
$sousetat['intervalle_fin']=5;
// entete
$sousetat['entete_flag']=1;
$sousetat['entete_fond']=1;
$sousetat['entete_hauteur']=7;
$sousetat['entete_fondcouleur']="255-255-255";
$sousetat['entete_textecouleur']="0-0-0";
```

(suite sur la page suivante)

(suite de la page précédente)

```
// tableau
$sousetat['tableau_bordure']=1;
$sousetat['tableau_fontaille']=10;
// bordure
$sousetat['bordure_couleur']="0-0-0";
// sous etat fond
$sousetat['se_fond1']="243-246-246";
$sousetat['se_fond2']="255-255-255";
// cellule
$sousetat['cellule_fond']=1;
$sousetat['cellule_hauteur']=7;
// total
$sousetat['cellule_fond_total']=1;
$sousetat['cellule_fontaille_total']=10;
$sousetat['cellule_hauteur_total']=15;
$sousetat['cellule_fondcouleur_total']="255-255-255";
// moyenne
$sousetat['cellule_fond_moyenne']=1;
$sousetat['cellule_fontaille_moyenne']=10;
$sousetat['cellule_hauteur_moyenne']=5;
$sousetat['cellule_fondcouleur_moyenne']="212-219-220";
// nombre d enregistrement
$sousetat['cellule_fond_nbr']=1;
$sousetat['cellule_fontaille_nbr']=10;
$sousetat['cellule_hauteur_nbr']=7;
$sousetat['cellule_fondcouleur_nbr']="255-255-255";
```

3.5.5 lettretype.inc.php

parametres

```
// general
$variable='&'; // nouveau
// $variable=chr(163); // compatibilite openmairie <4
// parametres
$lettretype['orientation']='P';
$lettretype['format']='A4';
// logo
$lettretype['logo']='logopdf.png';
$lettretype['logoleft']='58';
$lettretype['logotop']='7';
// titre
$lettretype['titreleft']='41';
$lettretype['titretop']='36';
$lettretype['titrelargeur']='130';
$lettretype['titrehauteur']='10';
$lettretype['titrefont']='helvetica';
$lettretype['titreattribut']='B';
$lettretype['titretaille']='15';
$lettretype['titrebordure']='0';
$lettretype['titrealign']='C';
// corps
$lettretype['corpsleft']='7';
$lettretype['corpstop']='57';
$lettretype['corpslargeur']='195';
```

(suite sur la page suivante)

(suite de la page précédente)

```
$lettretype['corpshauteur']='5';
$lettretype['corpsfont']='helvetica';
$lettretype['corpsattribut']='';
$lettretype['corpstaille']='10';
$lettretype['corpsbordure']='0';
$lettretype['corpsalign']='J';
```

3.6 Les vues

Les vues ne sont utilisables qu'avec postgresql et elles apparaissent dans la grille d'affichage des objets du générateur dans le fieldset « vues ».

Il est possible d'utiliser les vues pour faire des formulaires, états, requetes mémorisés … de la même manière que les tables sauf au niveau de la mise a jour (il faut paramétrer la vue de manière particulière)

Il est possible d'utiliser dblink pour créer une vue dans une base externe.

Les vues ont été initiées dans la version 4.1.0.

3.6.1 vue interne

creation d'une vue en sql

```
CREATE OR REPLACE VIEW om_administrateur AS
SELECT om_utilisateur.om_utilisateur AS om_administrateur, om_utilisateur.nom,
om_utilisateur.login, om_utilisateur.om_profil
FROM om_utilisateur
WHERE om_utilisateur.om_profil::text = '5'::text;
```

3.6.2 vues externes avec dblink

3.6.2.1 install dblink

ubuntu : pakage postgresql 8.X ou 9.X contrib dans synoptic

Dans la base cible, l'installation des fonctions dblink se fait en executant la requête var/share/postgresql/8.X ou 9.X/contrib/dblink.sql

test sql

```
SELECT dossier,nature FROM dblink('dbname=openfoncier','SELECT dossier,nature FROM
    dossier where nature = \'PC\'') as (dossier varchar(11), nature char(2))
```

Attention, il vaut mieux ne pas mettre les chaines de connexion dans les fichiers de parametrage openMairie

3.6.2.2 Creation de vue externe

en utilisant dblink

```
-- creation de vues sur un même serveur

CREATE VIEW openboisson_etablissement AS
   SELECT *
        FROM dblink('dbname=openboisson','SELECT etablissement, raison_sociale FROM
        etablissement') as (openboisson_etablissement integer ,
                            raison_sociale varchar(30));
```

3.6.3 Problème à régler dans l'utilisation d une vue externe

— il faut utiliser les fonctions d'encodage de pgsql si les 2 bases n'ont pas le même encodage
— il faut utiliser une sequence externe ou interne en insert
— il faut vérifier la cle secondaire dans la base ou schéma d'origine
— attention : la creation de vue non opérationnelle fait dysfonctionner le générateur qui fait appel au catalogue de vue : select viewname from pg_views

CHAPITRE 4

L'information geographique

Il est necessaire que l'API openLayers soit dans le framework :

lib/openlayers

4.1 Principe

Il est proposé dans ce chapitre de decrire le module tab_sig.php qui permet la geo localisation d'objet dans openMairie

Ce module est accessible dans la version 4.01 du framework et il est utilisé dans les applications openMairie suivantes

```
openmairie domainepublic
openmairie foncier
openmairie debitboisson
openmairie cimetiere
openmairie circulation
openmairie taxepub
openmairie triSelectif
openmairie adresse postale
openmairie openelec (projet)
openmairie resultat (projet)
openmairie dia (projet)
openmairie erp (projet)
```

L'objectif de tab_sig_map est de permettre une saisie le plus souvent automatique par un point, ligne, multiligne, polygone, multipolygone. Cette saisie est stockée dans la base métier postgresql. Elle est affichée sur des fonds existants sur internet : google sat, openStretmap ou bing (pour l instant) en utilisant le composant javascript openLayers

Il n'est donc pas nécessaire de disposer d'un SIG pour utiliser tab_sig.php.

Le format de stockage des données pgsql est celui de l'OGC et il est accessible aux clients libres où propriétaires qui respectent ce format (QGIS, GRASS, VEREMAP . . . pour les clients libres)

4.1.1 Géo localisation automatique

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 2 programmes (voir paragraphe sur le geocodage) :
- 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 script tab_sig.php permet de saisir manuellement le point.

4.1.2 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 a 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 datas 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. (voir paragraphe layers)

tab_sig.php permet

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

dans la version 4.2.0, tab_sig permet aussi

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

4.1.3 Paramétrage de la carte

Le paramétrage général (contenu dans scr/tab_sig.php) des cartes est modifiable dans dyn/var_sig.inc

```
// *** parametre de tab_sig.php ***
// generer une cle pour le site : http://code.google.com/intl/fr/apis/maps/signup.html
$cle_google = "";
$fichier_jsons="json_points.php?obj=";
$fichier_wkt="wkt_point.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;
```

(suite sur la page suivante)

(suite de la page précédente)

```
$zoom_bing=4;
// popup data contenuHTML
$width_popup=200;
$cadre_popup=1;
$couleurcadre_popup="black";
$fontsize_popup=12;
$couleurtitre_popup="black";
$weightitre_popup="bold";
$fond_popup="yellow";
$opacity_popup="0.7";
// image localisation maj ou consultation
$img_maj="img/punaise.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

// *** parametres d om_sig_map.class.php, om_sig_wms.class.php
$contenu_etendue[0]= array('4.5868,43.6518,4.6738,43.7018',
                           '4.701,43.3966,4.7636,43.4298',
                           '4.71417,43.64,4.72994,43.65166',
                           '4.72345,43.55348,4.73134,43.55932',
                           '5.2094,43.4136,5.3345,43.4759'
                           );
$contenu_etendue[1]= array('agglomeration',
                           'salin de giraud',
                           'raphele',
                           'Mas thibert',
                           'vitrolles'
                           );
$contenu_epsg[0] = array("","EPSG:2154","EPSG:27563");
$contenu_epsg[1] = array("choisir la projection",'lambert93','lambertSud');
```

4.2 objet map

ce chapitre propose de décrire l'utilisation de l'objet map d'openLayers dans tab_sig.php.

Cet objet permet de définir

- le div ou la carte sera affichée (dans tab_sig.php la carte s'affiche avec le div : map-Id)
- les options de la carte et les controles affichés

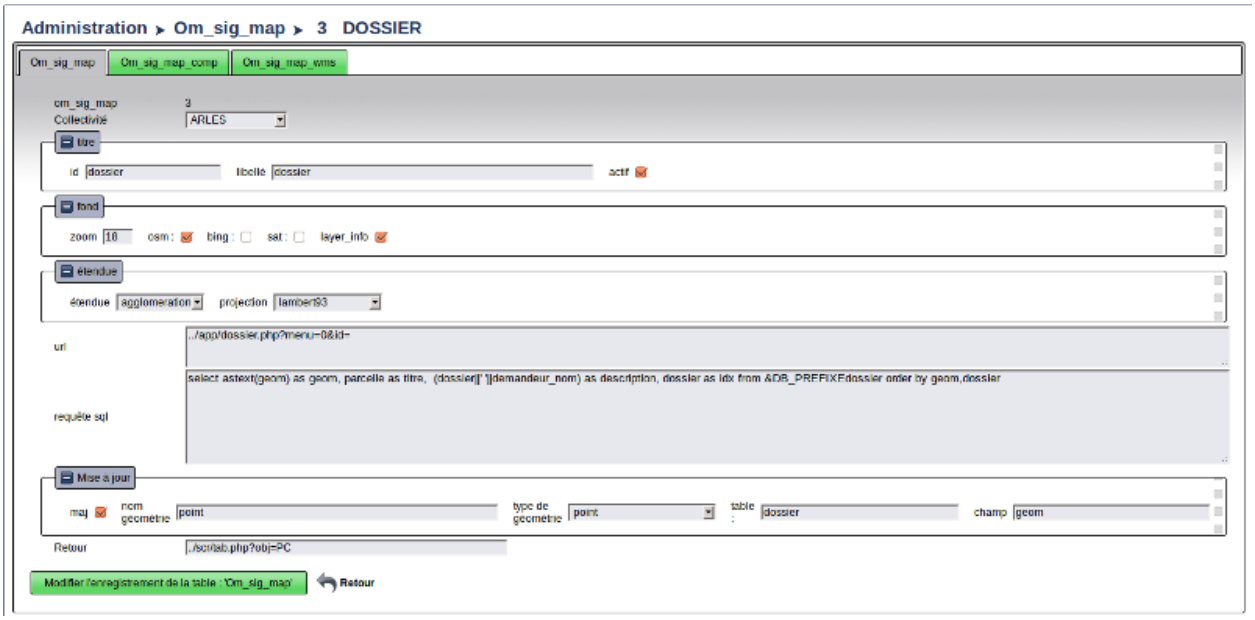
4.3 afficher les layers

ce chapitre propose de décrire l'utilisation de l'objet layers d'openLayers dans tab_sig.php.

Dans le lien, il est possible de définir

```
- la  carte a afficher suivant l'id : ?obj=   Obligatoire
- le fond affichable par défaut : sat, bing, osm : &fond =
- l'étendue : &etendue =
- l enregistrement à modifier : &idx=
```

Les cartes sont paramétrées dans om_sig_map (menu administration)



Il est possible de copier une carte et de paramétrer les champs suivants :

```
- id : identifiant unique (obligatoire)
- libelle
- fonds a afficher et data (osm, bing, sat(google))
- étendue et epsg (voir sig/var_sig_point.inc)
- url (qui pointe sur la fiche ou le formulaire de saisie)
- requete sql qui affiche les données json et qui doit désigné :
    le titre
```

(suite sur la page suivante)

(suite de la page précédente)

```
    la description
    l idx
- la mise a jour si oui,
            le champ géometrique et la table maj,
            le type de géometrie et le nom de la couche openLayers (version 4.2.0)
- le retour de la carte
```

Ces cartes sont possibles d'intégrer dans des menus, dans un

Dans tab_sig.php, il y a 3 types de layers :

- les fonds de cartes existants sur internet (base layers)
- les données issus de postgresql (overlays)
- les données wms (overlay)

4.3.1 Les fonds

Il est proposé les fonds suivants :

osm : openstreetmap

sat : satelite google

bing : satellite microsoft

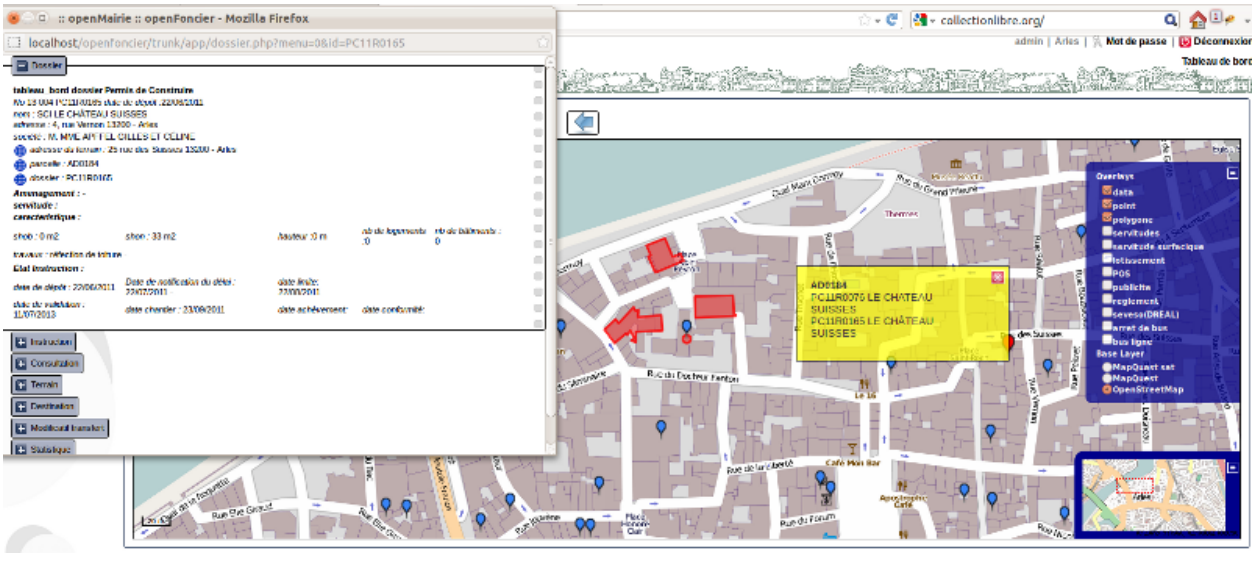
4.3.2 Les datas

Information de la carte : layer_info

Cette couche fait appel à sig_json.php

Il est possible de faire appel a un autre script (voir dyn/var_sig.inc)

La requête pgsql est paramétrée dans la table om_sig_map et doit définir les champs geom, titre, description et texte.

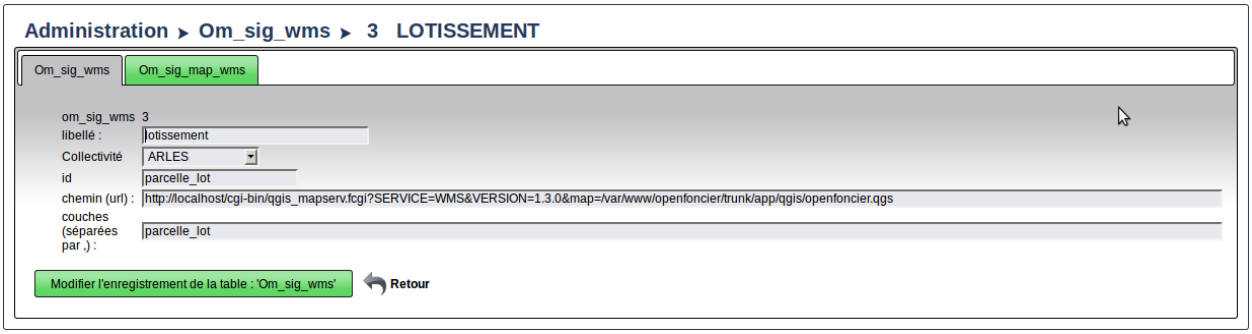


sig_json.php présente tous les enregistrements d'un même point (même géom) sur un seul popup

En effet, il est constitué un popup lorsque l on clique sur l objet et donne la possibilité à un accès URL parametrée dans om_sig_map

4.3.3 Les flux wms

Le paramètrage des flux wms est saisi dans om_sig_wms



il faut saisir

```
- libelle du champ
- la collectivité
- l'identifiant (il doit être unique pour chaque couche wms)
- le lien de la couche (http)
- les layers de la couches séparés par une virgule
```

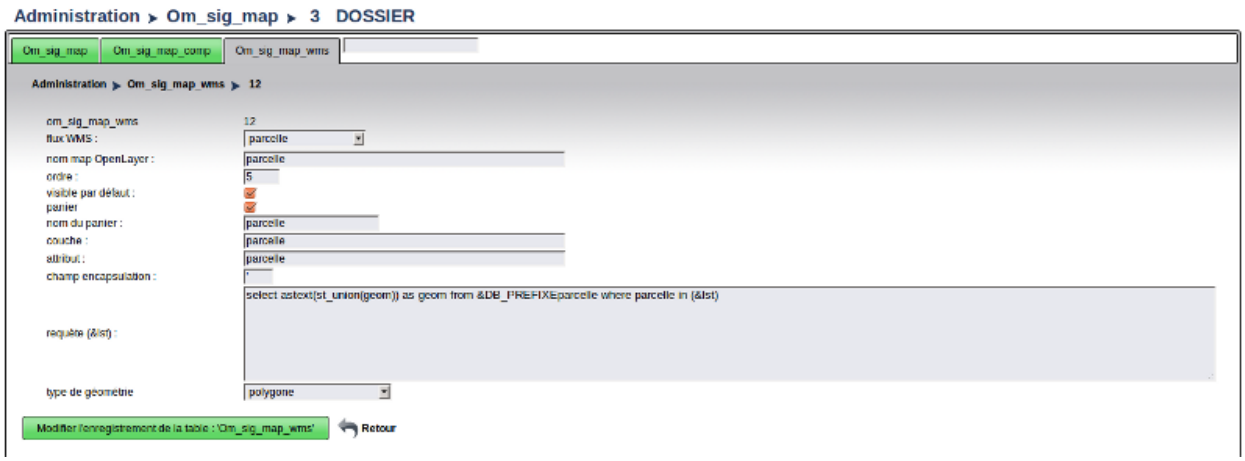
Exemple de lien avec qgis serveur

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

L'affectation des flux wms dans une carte est saisi dans om_sig_map_wms

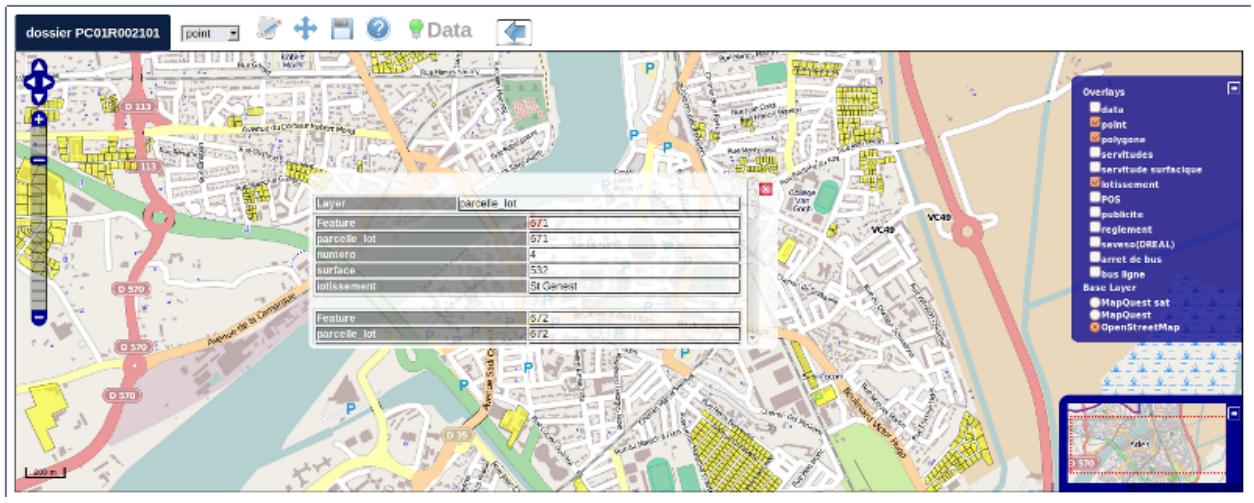
Il est saisi

```
le nom du flux wms
nom du layer sur la carte
l ordre d affichage
la visibilité par défaut (case à cocher)
```



Sur la carte ci dessous le flux wms est activé et affiche le lotissement (getMap)

En cliquant sur le lotissement, il est possible d'accéder aux données (getFeature)



4.3.4 La notion de pannier

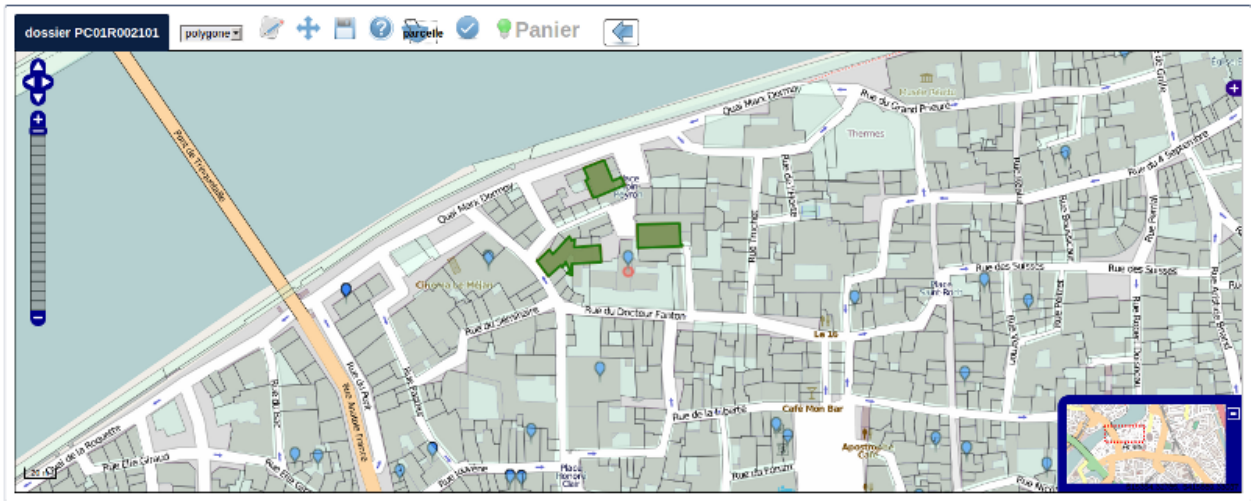
Le pannier permet de pouvoir stocker des géométries au travers de flux wms mais attention, la géométrie est récupérée dans une table ou une vue postgis (c'est pour l'instant une limite de la version 4.2.0)

exemple : openFoncier carte dossier :

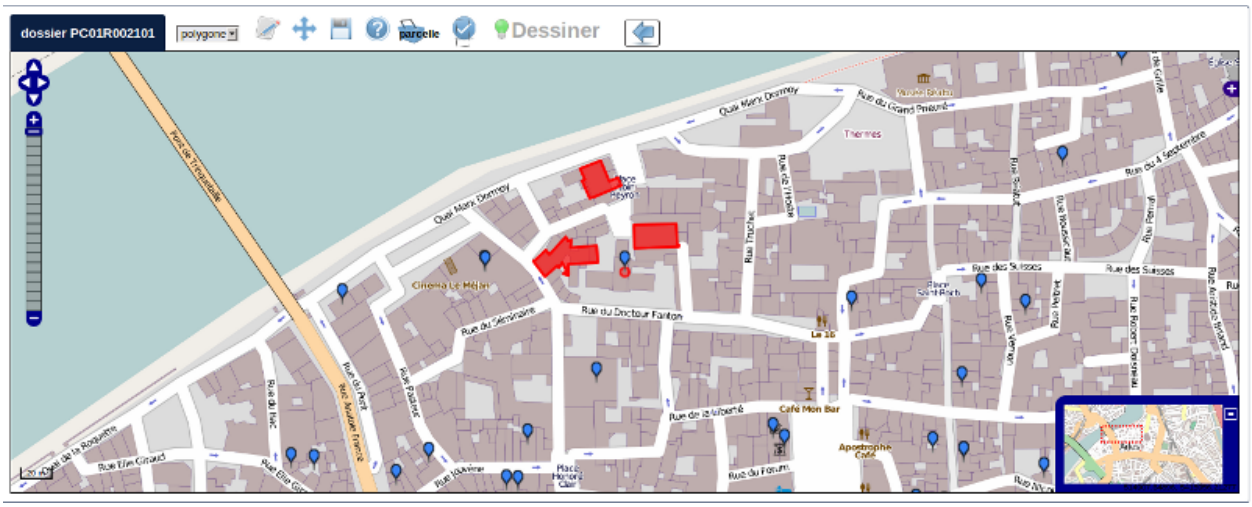
Il est proposé dans ce cas de stocker des polygones dans le pannier et de sauvegarder un multipolygone constitué de ces polygones récupérés dans le pannier

Choisir dans le select « polygone » ; L'etat est « dessinner »

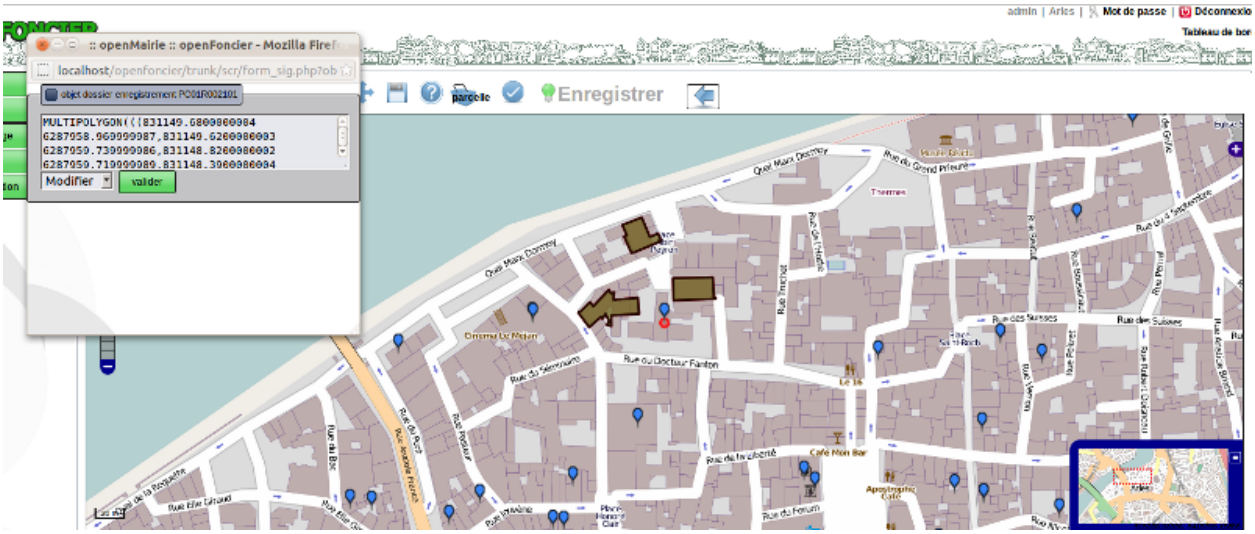
Il apparait le pannier « parcelle ». Sélectionner les parcelles en cliquant dessus (elles sont vertes)



Valider une fois les parcelles choisies (elles deviennent rouge)



Appuyer sur « enregistrer », l'état devient enregistrer



Cliquer sur le jeu de parcelles de votre choix (ce jeu devient vert clair)

Il peut y avoir un ou plusieurs panniers : exemple : parcelle, batiment. par contre la géométrie récupérée ne concerne qu une seule couche

la gestion de pannier se fait dans om_sig_map_wms

```
panier :          option pannier activé (Oui/non)          Exemple dossier/openFoncier :
pa_nom :          nom du pannier                           parcelle
pa_layer :        nom du layer pannier                     parcelle
pa_attribut:      attribut de la couche à récupérer        parcelle
pa_encaps:        caractère d'encapsuation (la ')          '
pa_sql:           requête de récupération                  select astext(st_union(geom))␣
 ↪as geom
                                                           from &DB_PREFIXEparcelle␣
 ↪where parcelle in (&lst)
pa_type_geometrie:  type de géométrie                      polygone
```

le script de gestion de pannier est : scr/sig_pannier.php

4.3.5 La géométrie à modifier : couche vectors :

Le chargement de la couche vectors se fait si dans la table om_sig_map, la case maj est activée.

La géométrie est récupérée par le script sig_wkt.php (appel a un script paramètrable dans var_sig.inc) et la carte est centrée sur la géométrie

Il est possible de :

- positionner manellement la géométrie
- déplacer la géométrie
- **enregistrer la géometrie** [selectionner la géométrie, le programme] form_sig.php est chargé en fenetre et permet de supprimer la géométrie (champ geometrique = null) ou modifier cette géométrie.

Les fonctions javascript et les controles sont activées suivant chaque état.

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

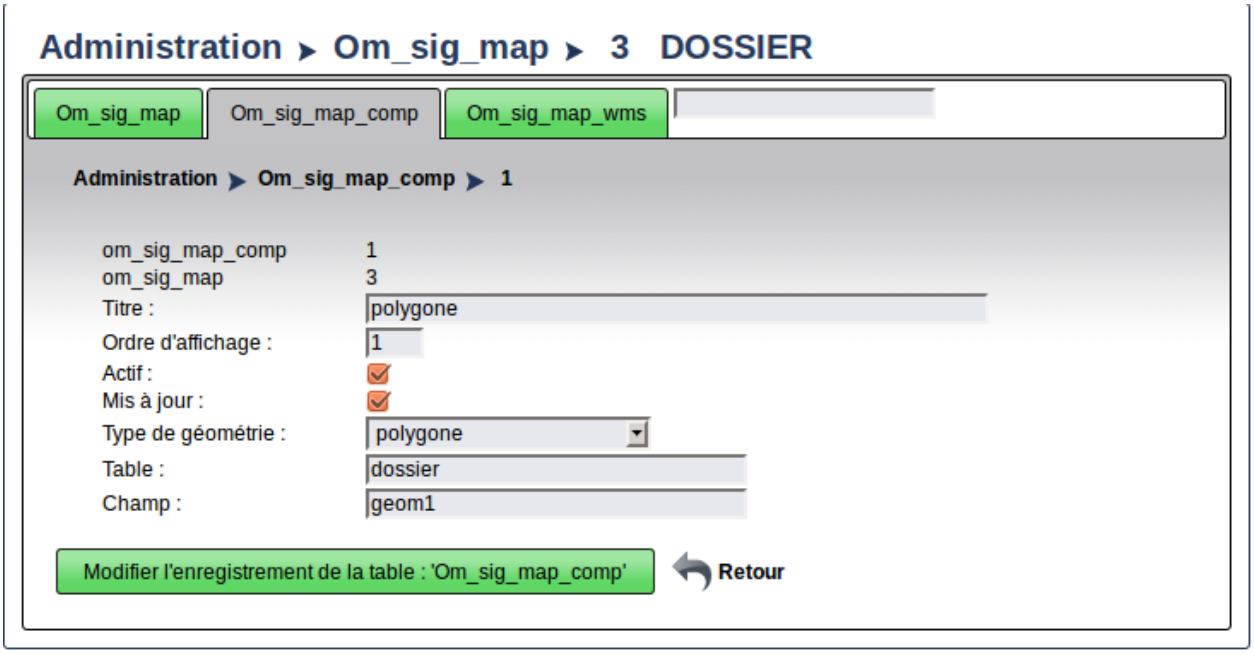
Dans dyn/form_sig_delete.inc.php, il est possible de paramétrer des post traitements de suppression

4.3.6 Les géométries complémentaires

cd so Il peut y avoir plusieurs géométries pour un même objet.

Elles sont saisies dans om_sig_map_comp

```
titre               polygone    nom de la nouvelle géométrie
ordre d affichage   1           ordre d'affichage dans le select
actif               coché       activé la nouvelle géométrie
Mise a jour         coché       autorisé la mise à jour
type de géométrie   polygone    polygone, point, ligne
table               dossier     table du champ géométrique
champ               geom1       champ géometrique concerné
```



Dans l exemple précédent, il apparait une fenêtre select ou l utilisateur a le choix entre une géométrie « point » et une géométrie « polygone » du fait de la mise en place d'une géométrie complémentaire.

4.4 format

ce chapitre se propose de decrire les divers formats de transfert avec openLayers tab_sig.php n'utilise que le format json et wkt.

4.4.1 wkt

triangle ESPG 4326

```
POLYGON
    (
        (
        -11.109377145767 51.70313000679,
        19.124997854233 19.35938000679,
        43.031247854233 44.67188000679,
        -11.109377145767 51.70313000679
        )
    )
```

4.4.2 geojson

Le format JSON correspond à la notation {paramètre : valeur, paramètre : valeur} de JavaScript, qui a l'avantage de ne pas nécessiter de conversion (« parsing »), puisqu'il est directement intégré par le moteur d'exécution JavaScript du navigateur

format 2 polygones

```
{
    "type": "FeatureCollection",
    "features": [
        {"type":"Feature",
         "id":"OpenLayers.Feature.Vector_1489",
         "properties":{},
         "geometry":
            {"type":"Polygon",
             "coordinates":
                [[[-109.6875, 63.6328125],
                  [-112.5, 35.5078125],
                  [-85.078125, 34.8046875],
                  [-68.90625, 39.7265625],
                  [-68.203125, 67.1484375],
                  [-109.6875, 63.6328125]
            ]]},
            "crs":
                {"type":"OGC",
                 "properties":{"urn":"urn:ogc:def:crs:OGC:1.3:CRS84"}}
                },
        {"type":"Feature",
        ...
        }
    ]
}
```

4.5 installation d'om_sig_map

Pour faire fonctionner tab_sig.php, il faut :
- postgis
- openlayers qui est de base dans le framework lib/openlayers

4.5.1 intsallation de postgis

sur UBUNTU installer les paquets postgis

```
- postgis
- postgresql-8.3.postgis
```

PostGIS ajoute à postgresql

```
- ses types de données
- ses fonctions,
- deux tables utilitaires : geometry_colums et spatial_ref_sys.
    geometry_colums sert à indiquer au logiciel quels sont les champs contenant des␣
↪types
    géographiques dans chacune des tables.
    spatial_ref_sys contient les paramètres des systèmes de projection  supportés
    et sert en interne au logiciel.
```

Verification de l install postgresql-postgis

```
postgre> select version()

"PostgreSQL 8.3.9 on i486-pc-linux-gnu, compiled by GCC gcc-4.3.real
(Ubuntu 4.3.3-5ubuntu4) 4.3.3"

postgre> show server_version

"8.3.9"

postgre> show data_directory

"/var/lib/postgresql/8.3/main"
```

version proj et geao

```
postgre > select postgis_full_version() ::

"POSTGIS="1.3.5" GEOS="3.1.0-CAPI-1.5.0" PROJ="Rel. 4.6.1, 21 August 2008"
USE_STATS (procs from 1.3.3 need upgrade)"
```

Voir paragraphe « outils »

4.5.2 Paramétrage d'une base avec postgis

- creer la base openmairie (si elle n'est pas deja creee)
- postgre> create language « plpgsql »
- executer (version postgis <1.5) la requete lwpostgis.sql -> fonction postgis ou executer (version postgis >= 1.5) la requete /usr/share/postgresql/8.3/contrib/postgis-1.5/postgis.sql
- executer spatial_ref_sys .sql qui remplit la table de donnees spatial_ref_sys

— VERIFICATION : les tables suivantes sont presentes

```
* table geometry_columns : index des geometries (vide)
* table spation_ref_sys : liste des references spatiales (3162 lignes environ)
```

— executer les scripts d'initialisation de la base

```
* data/pgsql/init.sql
* data/pgsql/initsig.sql
* data/pgsql/initsig_data.sql (optionnel) jeu de donnees
```

verification des bases : liste des bases en console

```
$ psql -l

        Liste des bases de données
      Nom      | Propriétaire | Encodage
---------------+--------------+-----------
 alaska        | postgres     | UTF8
 cadastre      | postgres     | SQL_ASCII
 opencimetiere | postgres     | SQL_ASCII
 openelec      | postgres     | SQL_ASCII
 openelec1     | postgres     | SQL_ASCII
 openerp       | postgres     | SQL_ASCII
 openfoncier   | postgres     | SQL_ASCII
 openmairie    | postgres     | SQL_ASCII
 postgres      | postgres     | UTF8
 sig           | postgres     | SQL_ASCII
 template0     | postgres     | UTF8
 template1     | postgres     | UTF8
 xx            | postgres     | SQL_ASCII
(13 lignes)
```

acces a opencimetiere

```
$ psql opencimetiere

Bienvenue dans psql 8.3.11, l'interface interactive de PostgreSQL.
    \h pour l'aide-mémoire des commandes SQL
    \? pour l'aide-mémoire des commandes psql
    \g ou point-virgule en fin d'instruction pour exécuter la requête
    \q pour quitter
```

exemple de selection des colones geometriques de la base « odp »

```
$ psql odp -Atc "SELECT f_table_name|| '('||type||')' from geometry_columns"

odp(POINT)
```

4.5.3 partager un serveur postgresql

se connecter sur le serveur postgresql en ssh

```
$ ssh numeroIP
```

autoriser les IP externes a se connecter

```
etc/postgresql/8.x/main/pg_hba.conf
rajouter la ligne des postes ayant acces
$ sudo nano pg_hba.conf
    # toutes les IP commencant par 10.1
    host    all all 10.1.0.0/16 trust
    # permis pour IP 10.1.30.10
    host    all all 10.1.30.10/32   trust
```

configurer le port 5432 comme port d ecoute

```
etc/postgresql/8.x/postgresql.conf
$ sudo nano postgresql.conf
# ecoute sur le port 5430 toutes adresses
listen_adresses='*'

$ netstat -lpn

tcp  0  0 0.0.0.0:5432   0.0.0.0:*               LISTEN      -
```

changer le mot de passe postgresql

```
$ sudo su - postgres
postgres@ubuntu-1011015:~$ psql
postgres=# alter user postgres  with password 'postgres'
postgres-# \q
```

connexion distante sur pgadmin nom : serveurdev hote : 10.1.0.12 util : postgres pwd : postgres

4.5.4 optimisation composant openLayers

construire un OpenLayers.js compresse dans le repertoire build

```
$ cd buill
$ python build.py
```

le fichier fait 800 ko au lieu de 3 Mo

— compression lite

```
$ python build.py lite.cfg
le fichier fait 120 ko
regarder dans le fichier "lite" les fichiers qui sont inclus
et éventuellement le compléter
```

4.6 postgis

ce chapitre propose de décrire les possibilités d'utilisation de postgis dans openMairie.

4.6.1 principes

Il est proposé un renvoi sur la documentation française.

http://postgis.refractions.net/documentation/manual-1.3/ch06.html

Les requêtes utilisant des fonctions potgis peuvent être implémentées dans « reqmo »

Il sera proposé un lien sur un tutorial utilisant les fonctions postgis.

4.6.2 base et schéma

Il est noté que les applications openMairie peuvent s'installer dans un schéma.

Les tables et fonctions postgis sont alors accessible dans le schéma public.

4.7 geocodage

Ce chapitre est consacré au problème de géocodage.

Il est propose un géocodage interne (sgbd adresse en reseau interne) ou un geocodage externe

Il convient de regarder les termes de licences concernant les API externes non libres (mapquest/osm) afin de s'assurer de bien respecter les obligations de l'autorisation gratuite.

Un document décrivant les contraintes juridiques et techniques de l'utilisation des API est accessible au lien suivant :

http://www.openmairie.org/communautes/groupe-de-travail-sig-adullact/comparatif-api.pdf/view

Le parametrage se fait dans le fichier sig/var_adresse_postale.php

4.7.1 var_adresse_postale.inc

paramètrage général

```
$longueurRecherche=1;
```

adresse postale stockée sur une base dans le réseau interne

```
// table et champs de la requete adresse postale ou l information doit etre recupere
$t="adresse_postale";                       // table adresse postale
$t_voie = "rivoli";                         // code adresse
$t_numero="num_voi";                        // numero dans la voie
$t_complement="suf_voi";                    // suffixe (bis, ter ...)
$t_geom="the_geom";                         // geometry point(X,Y)
$t_adresse="(typevoie ||' '||nomvoie)";     // libelle de l adresse
$t_quartier="id_parc";
// *** a voir
$t_cp='';                                   // nom champ cp
$t_ville='';                                // nom champ ville
$t_insee='';                                // nom champ insee

// champ du formulaire ou l adresse est saisi pour retour du point de geolocalisation
$f_numero='numero_voie';        // nom champ du numero dans la voie
$f_voie='voie';                 // nom champ du code de la voie (rivoli)
$f_complement='complement';     // nom champ du complement de numero
$f_geom='geom';                 // nom champ geometrique point(X,Y)
$f_libelle='libelle_voie';      // nom champ libelle de la voie
// *** a voir
$f_cp='';                       // nom champ cp
$f_ville='';                    // nom champ ville
$f_insee='';                    // nom champ insee
```

Cas ou la table d'adresse est stockées dans une autre base (voir parametrage framework de database.inc.php) :

```
$db_externe='Oui';  //  Oui : base externe
$dsn_externe= array(
    'title'  =>"base locale des adresses de l'IGN",
    'phptype'  => "pgsql",
    'dbsyntax' => "pgsql",
    'username' => "postgres",
    'password' => "postgres",
    'protocol' => "tcp",
    'hostspec' => "localhost",
    'port'     => "5432",
    'socket'   => "",
    'database' => "ignlocal",
    'formatdate'=> "AAAA-MM-JJ",
    'schema'  => "public",
    'prefixe'  =>""
);
$db_option_externe=array('debug'=>2,
    'portability'=>DB_PORTABILITY_ALL);
```

Géolocalisation par accès à un API externe

```
// variables par defaut cp et ville si non renseignées dans le formulaire
// pour recherche
$cp="13200";
$ville="Arles";
$pays = ""; // a voir
// epsg de transformation pt adresse postale dans la base en cours
$epsg= "EPSG:27563";
// acces au script adresse_postale externe
$adresse_interne="Oui";
$google="Oui"; // google
$bing="Oui";   // bing
$osm="Oui";    // mapquest
```

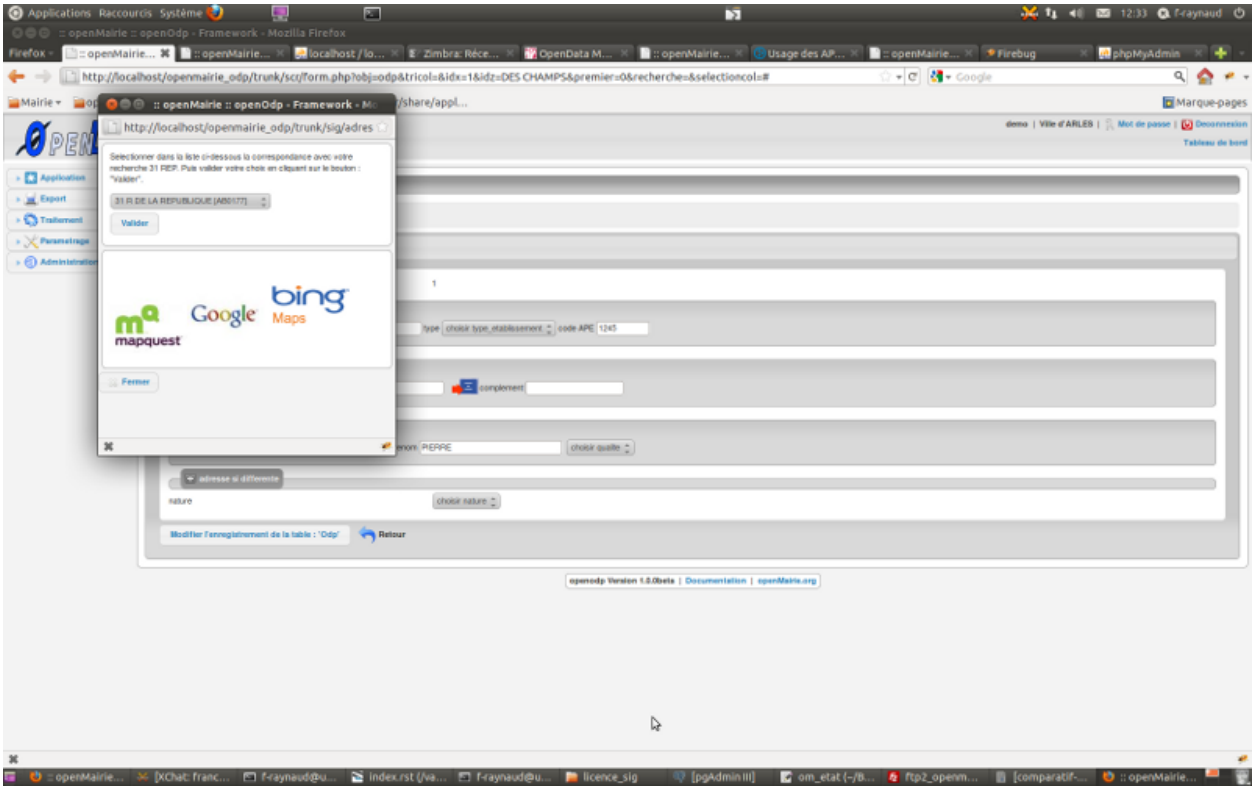
le parametre $adresse_interne à Oui permet de consulter une adresse stiockée dans le réseau interne sur la même base, ou sur une base différente (voir plus haut)

Ensuite 3 API peuvent être initialisés : google, bing et mapquest

4.7.2 Mise en oeuvre dans un formulaire d'un bouton de la geolocalisation

La géolocalisation se fait sur la base du script

```
sig/adresse_postale.php
qui fait appel suivant le paramétrage à :
    adresse_postale_bing.php
    adresse_postale_google.php
    adresse_postale_mapquest.php
```



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 geographique (lattitude,␣
↪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␣
↪géometrique 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." "._("le point trouvé par l'API est␣
↪sauvegardé")." ".
            $this->table." ".$id;
        }
    }
```

4.8 qgis_server

Pour gérer les flux wms et wfs, il a été implémenté qgis_server

Ce chapitre propose quelques principes d'utilisation de qgis server

4.8.1 Installation de QGIS server sur UBUNTU

Article d'origine http://geotribu.net/node/286 modifié suite teste le 07/06/2012 avec la version 1.8 de qgis server

Installation de qgis server

La communication entre QGIS Mapserver et notre serveur Web s'appuie sur le protocole CGI/FCGI

```
$ sudo apt-get install libfcgi-dev
```

l'installation de QGIS Mapserver se fait en utilisant le dépôt ubuntugis

```
$ sudo apt-get install qgis-mapserver
```

Il est créé un répertoire /usr/lib/cgi-bin/, le fichier qgis_mapserv.fcgi qui interprete les requêtes WMS et retourne ensuite sous forme d'imagettes (flux wms).

4.8.2 parametrage de qgis server

il faut créer un nouveau dossier dans ce répertoire cgi-bin pour chaque projet qgis (si on veut un acces externe hors poste de production localhost).

exemple pour le projet formation

```
usr/lib/cgi-bin$ sudo mkdir formation
```

Ensuite, il faut créer trois liens symboliques.

Le premier pointant vers script qgis_mapserv.fcgi, le second vers votre projet QGIS et enfin le dernier vers le fichier wms_metadata.xml. On peut personnaliser ce dernier fichier en y ajoutant les différentes informations concernant le producteur de la donnée (nom de l'organisme, nom du référent...).

Ceci permet d'acceder en wms avec openlayers avec une IP externe (à vérifier)

```
/usr/lib/cgi-bin/formation$ sudo ln -s /var/www/projet/formation/qgis/formation.qgs .
```

pour l'acces externe, ces deux liens ne servent pas mais ils sont importants pour la fonction « getCapabilities »

```
/usr/lib/cgi-bin/formation$ sudo ln -s ../qgis_mapserv.fcgi .
/usr/lib/cgi-bin/formation$ sudo ln -s ../wms_metadata.xml .
```

4.8.3 requetes WMS / WFS

tab_sig.php utilise dans les requêtes

```
getMap
getFeature
```

4.8.4 paramétrage de vues

— regarder la table "geometry_columns" » ne soit pas cochée (option editer)
— modifier la clé primaire dans la liste des tables de connexion

4.9 Mapserver

mapserver n est pas utilise et il a été préféré qgis_server dans les deploiements openMairie

Ce chapitre propose quelques principes de map server

4.9.1 Principes

MapServer permet de générer des cartes à partir de données diverses et de fichiers de configuration, qui contiennent les paramètres décrivant la façon dont les données doivent être présentées, les mapfiles.

4.9.2 Installation UBUNTU

Verifiez que les depots Universe et Multiverse font partie de vos sources de mise a jour. installer les paquets suivants (obligatoire)

- cgi-mapserver
- mapserver-bin
- mapserver-doc

creer un repertoire avec droit d ecriture pour www.data pour stocker les images crees par mapinfo var/www/tmp/ pour ubuntu ou changer le chemin de ce repertoire dans les openmairie_cimetiere/sig/map/xxxx.map

```
WEB
    IMAGEPATH "/var/www/tmp/"
    IMAGEURL "/tmp/"
END
```

- ATTENTION Verifier le chemin dans openmairie_cimetiere/sig/var.inc qui doit correspondre a votre installation

4.10 outils

ce chapitre propose de décrire de manière sommaire les outils geographiques pouvant être utiles à l extension d'om_sig

```
GEOS pour le calcul geometrique
GDAL pour les rasters
OGR pour l'interrogation des données
PROJ pour la projection
```

GEOS et PROJ sont utilisés par postgis et mapserver.

Voir exemple d'utilisation : http://zoo-project.org/

4.10.1 PROJ pour les projections

Version de proj

```
$ proj -- version
```

la table des projection est dans usr/share/proj/epsg

```
# Google
<900913> +proj=merc +lon_0=0 +k=1 +x_0=0 +y_0=0 +ellps=WGS84 +datum=WGS84 +units=m␣
↪+no_defs<>
```

postgis utilise proj et la table ref_spatial_sys

4.10.2 GDAL pour les rasters

installation

```
$ sudo apt-get install gdal-bin
$ gdalinfo --formats

dependance de gdalinfo
$ ldd usr/bin/gdalinfo
$ apt-get install lib-gdal1-dev

pour faire des tuiles
apt-get install python_gdal
gdap to tile ?


format rasters

Origine : point bas gauche
largeur : pixelsize H
longueur : pixelsize V
taille du pixel
pixel : x,y + altitude (z)
```

Exemple d'utilisation de gdal pour merger deux images

```
$ gdal_merge.py -o srtm_location.tif srtm_37_04.tif srtm_38_04.tif
$ gdaldem hillshade srtm_location.tif shade.tif -z 5 -s 111120 -az 90
```

http://gdal.org/ogr/index.html

4.10.3 OGR pour l interogation de tout type de format

liste des format accessibles parogr

```
  $ ogr2ogr --formats | grep d/w ou $ ogr2ogr --formats ::

-> "ESRI Shapefile" (read/write)
-> "MapInfo File" (read/write)
-> "TIGER" (read/write)
-> "S57" (read/write)
-> "DGN" (read/write)
-> "Memory" (read/write)
-> "BNA" (read/write)
-> "CSV" (read/write)
-> "GML" (read/write)
-> "GPX" (read/write)
-> "KML" (read/write)
-> "GeoJSON" (read/write)
```

(suite sur la page suivante)

(suite de la page précédente)

```
-> "Interlis 1" (read/write)
-> "Interlis 2" (read/write)
-> "GMT" (read/write)
-> "SQLite" (read/write)
-> "ODBC" (read/write)
-> "PostgreSQL" (read/write)
-> "MySQL" (read/write)
-> "Geoconcept" (read/write)

  $ ogrinfo --formats
  $ history | grep odp ogr

  acces au sgbd postgres base "odp"
  $ ogrinfo -so "PG:dbname=odp"
  $ ogrinfo -so "PG:dbname=odp" odp | less

  acces a un fichier shape ::
  $ ogrinfo -so ./natural.shp

  faire une requete sur un shape
  $ ogrinfo -sql "SELECT type from natural" ./natural.shp  ogrinfo -so ./natural.shp

  requete fabriquant un shape ::
  $ ogr2ogr -sql "SELECT ST_Buffer(geom,10),ville,voie,complement from odp" test_data.
↪shp "PG:dbname=odp"
  $ ogr2ogr -sql "SELECT ST_Buffer(geom,1),ville,voie,complement from odp" test_data2.
↪shp "PG:dbname=odp"
```

http://dl.maptools.org/dl/php_ogr/php_ogr_documentation.html

4.10.4 GEOS

calcul geometrique (voir requete postgis)

4.11 data_sig

ce chapitre propose de décrire la récupération de données SIG nécessaires à la mise en oeuvre des scripts SIG internes openMairie.

4.11.1 Saisir le périmètre de sa commune

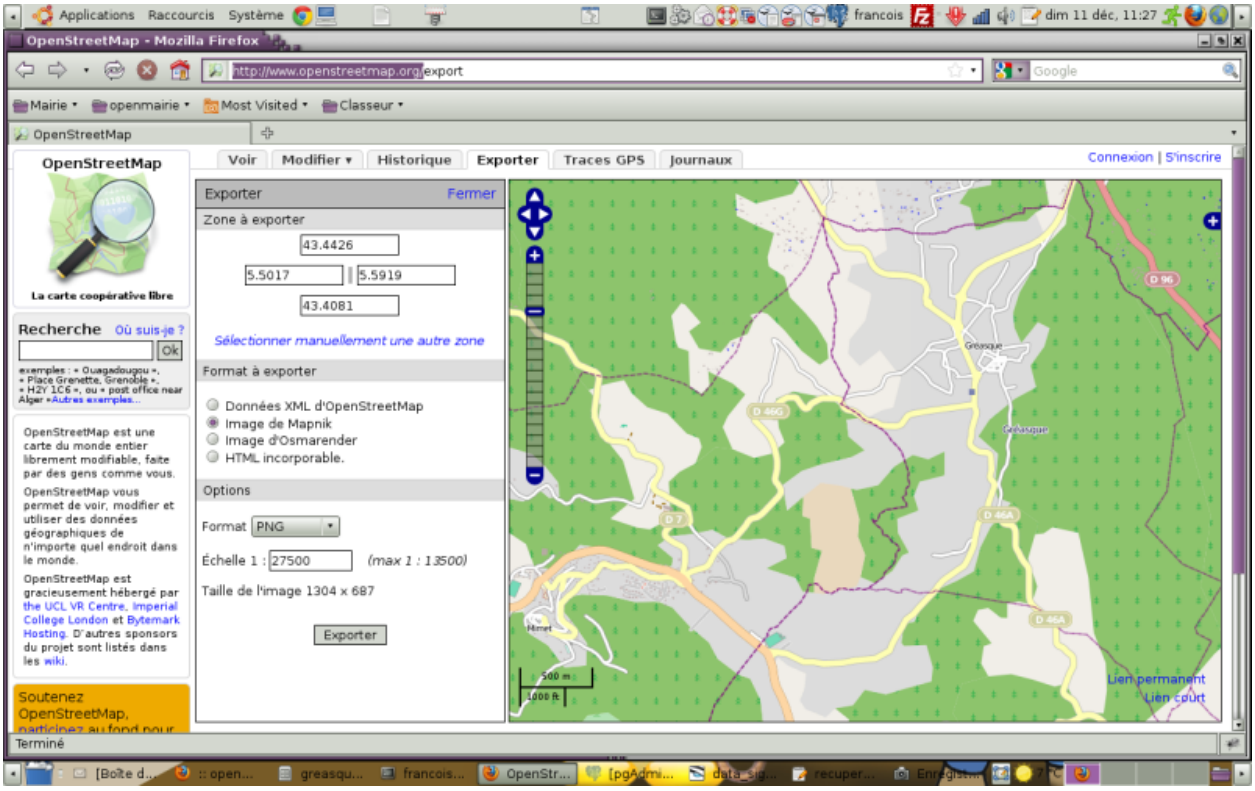
Il s'agit d'adapter ses cartes au périmètre de sa commune

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 »



Dans le fichier dyn/var_sig.inc, modifier le tableau de variables avec les coordonnées de la manière suivante

```
$contenu_etendue[0]= array('5.5155,43.4081,5.5781,43.4426');
$contenu_etendue[1]= array('greasque');
```

Modifier les cartes de om_sig_point

4.11.2 Récupérer les données de l'IGN

Exemple de récupération de données « parcelle » (fichier shape du CRIGE PACA)

Les données de l'IGN sont fournies aux communes par département.

Insérer le fichier parcelle dans la base (exemple IGN)

```
shp2pgsql -s 2154 -I -D -W LATIN1 PARCELLE.SHP | psql ign
```

Il est créer dans la base « ign » une table parcelle décrite ci dessous

```
gid serial NOT NULL,
        numero character varying(4),
        feuille smallint,
        section character varying(2),
        code_dep character varying(2),
        nom_com character varying(45),
        code_com character varying(3),
        com_abs character varying(3),
        code_arr character varying(3),
        the_geom geometry
```

ainsi qu un enregistrement parcelle dans géometry_columns (postgis est obligatoire dans la base) et un index

Selectionner les parcelles de la commune concernée et inserer les dans une nouvelle table

```
insert into parcelle_greasque       (parcelle, section, commune, geom)
select      section||numero, section, code_dep||code_com, the_geom  from parcelle
    where code_com = '046';
```

Pour mettre à jour le champ surface de parcelle dans openfoncier

```
update parcelle set surface = round(cast(area2d(geom) as numeric), 2)
```

4.11.3 Recupération des données de la DGI

Les fichiers textes de la DGI sont dans un format récupéré dans le cadre de l application openCadastre qui reconstitue des tables postgresql.

Les fichiers gémétriques au format EDIGEO ne sont pas récupérés compte tenu de son format non standard et il est préféré utiliser les formats shape de l IGN

4.11.4 Recuperation de la base adresse de l'IGN

Le projet de la ville de Vitrolles openAdresseIGN inclu dans le projet openAdresse se propose de construire un référentiel adresse pour une commune sur la base des fichiers fournis par l IGN.

CHAPITRE 5

Le tableau de bord et les widgets

5.1 principe

Il est proposé dans ce chapitre de décrire le tableau de bord paramètrable pour les utilisateurs

5.1.1 paramétrage

l'accès au tableau de bord paramètrable 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().

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.

5.1.3 le tableau de bord paramètrable

Chaque utilisateur choisit ses widgets parmi ceux proposés dans l'application par l'administrateur. Il peut placer ses widgets où il veut dans son tableau de bord. Le paramétrage est conservé dans la table om_tdb

5.2 widget

Le widget est un petit script qui s'exécute dans le tableau de bord dans une fenêtre normalisée.

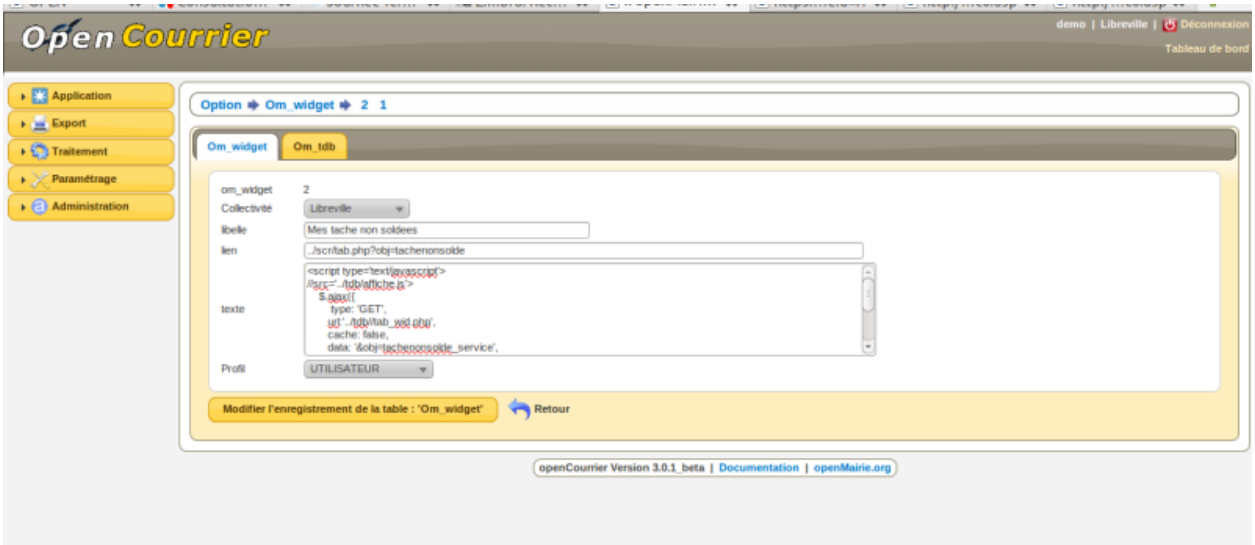
Le script peut faire appel à l'application en cours ou a une application externe. Nous avons mis quelques exemples dans les deux derniers paragraphes. Il est proposé d'abord de vous aider à créer les widgets.

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 apparaitra à l adition 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 ...
```

5.2.2 Les widgets internes

les liens sur les cartes (à mettre danbs 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,
```

(suite sur la page suivante)

(suite de la page précédente)

```
       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 different pour chaque div (ici aff3)

5.2.3 Les widgets externes

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>
```

(suite sur la page suivante)

(suite de la page précédente)

```
</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_fqcy=0
   &up_hl_background=FFFFFF&synd=open&w=200&h=100
   &title=
   &border=%23ffffff%7C3px%2C1px+solid+%23999999&output=js'>
</script>
```

Affichage de photos avec flick “r (appel javascript) :

```
<table><tr>
<div class='flick_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>
```

5.3 le tableau de bord paramétrable

ce paragraphe propose de décrire l’utilisation du tableau de bord paramétrable par utilisateur

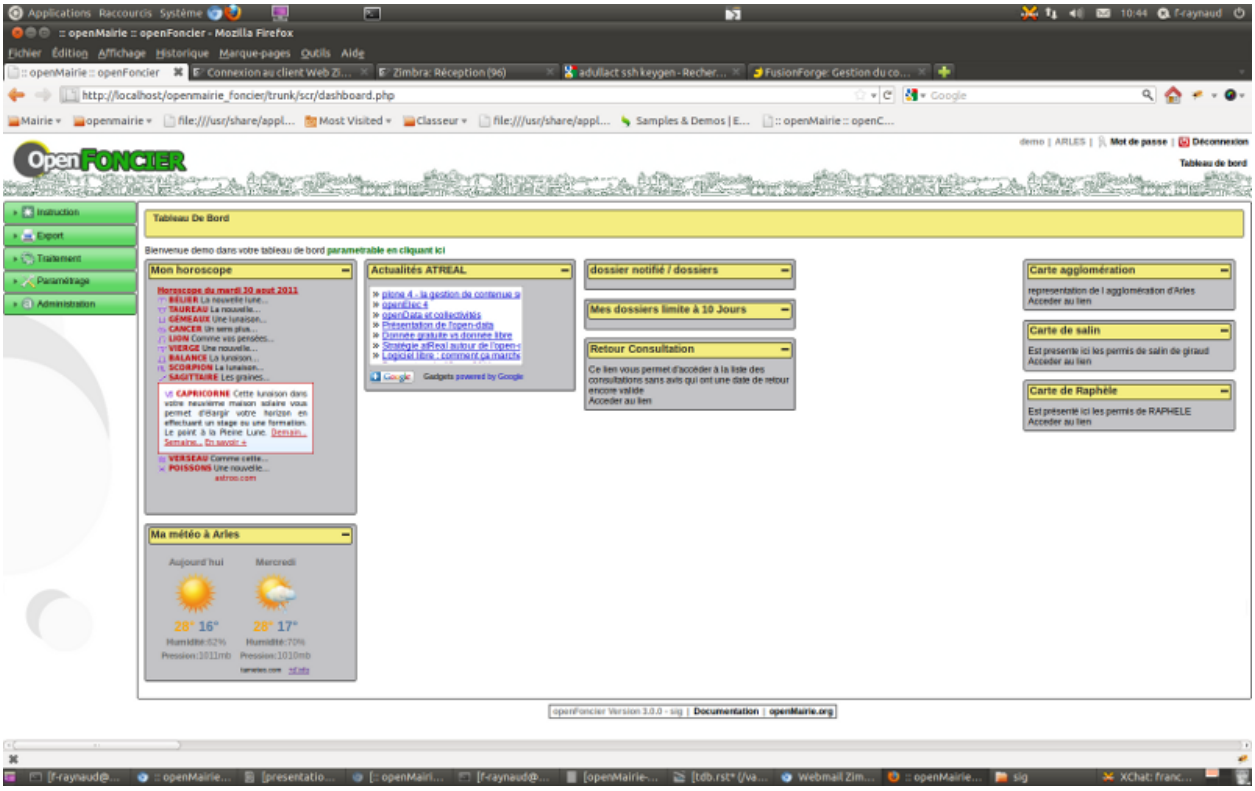
5.3.1 accès au tableau de bord

Le paramétrage se fait en cliquant sur le lien « paramétrer son tableau de bord »

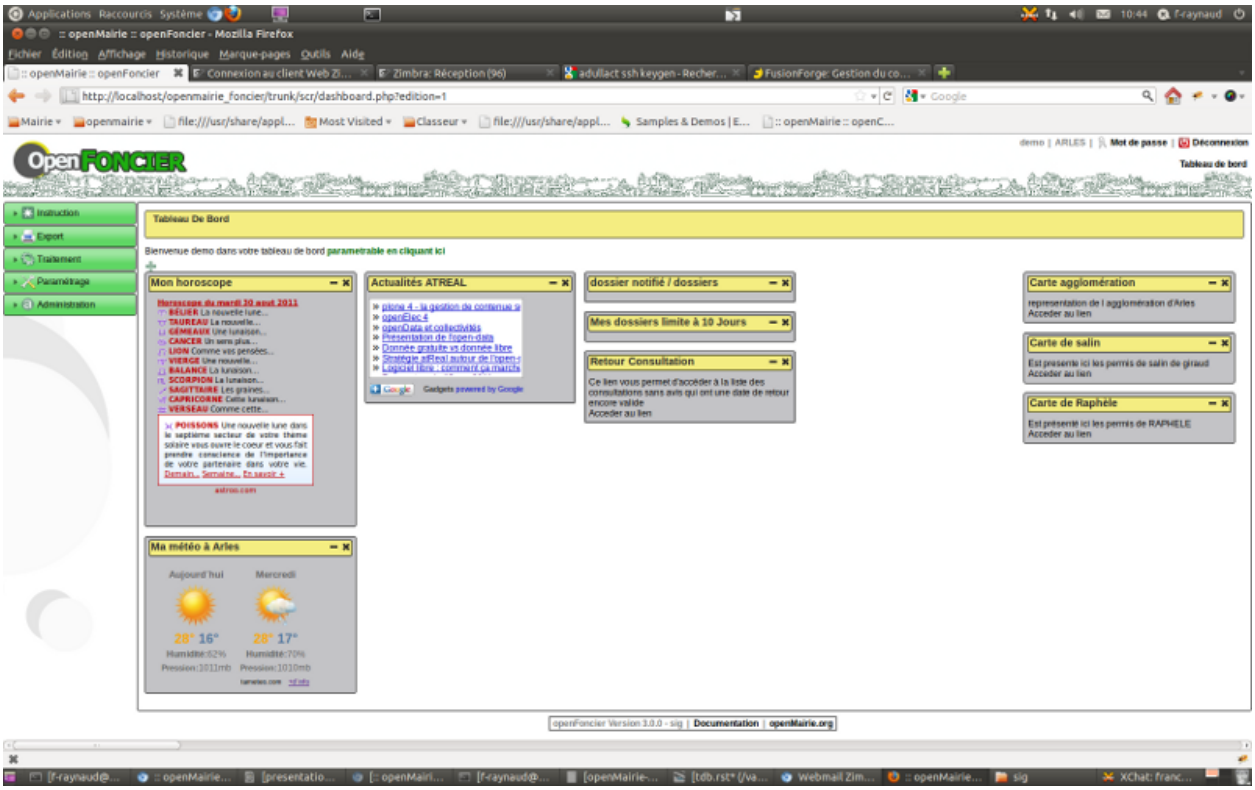
Il apparait alors

```
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



5.3.2 la table om_tdb

La table om_tbd comprend les champs suivants

```
om_tdb int(8) NOT NULL,  : numero d ordre
login varchar(40) NOT NULL, : login de l'utilisateur
bloc varchar(10) NOT NULL, : bloc ou colone (c1 ou c2 ou c3)
position int(8),   : position dans la colone
om_widget int(8) NOT NULL, : numero de widget dans om_widget
```

Attention, en cas de changement de login, un utilisateur perd ses paramètres

CHAPITRE 6

Règles et outils

6.1 Les règles de codage

La convention de codage openMairie s'applique à tout le code qui fait partie de la distribution officielle d'openMairie. La convention de codage permet de conserver un code consistant et de le rendre lisible et maintenable facilement par les développeurs openMairie.

6.1.1 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.

Il est recommandé que la longueur des lignes ne dépasse pas 75 à 85 caractères.

6.1.2 L'encodage des fichiers

L'encodage est un jeu de caractère qui permet de « transcrire des langues naturelles avec une représentation numérique pour chaque caractère » (wikipédia)

Pour la france, les jeux de caractères sont les suivants (source wikipédia)

```
* ASCII, standard de compatibilité, sans accent.
* Windows-1252, pour Windows, encore appelé CP1252 ou Ansinew
* ISO 8859-1, avant l'euro (latin-1 ou europe occidentale)
* ISO 8859-15, après l'euro (latin-9 ou latin-1)
* Unicode, voir aussi UTF-8 (reprend tous les caractères des jeux de code précédent␣
↪et se présente comme le standard).
```

La version 4.1.0 corrige les bugs en utf8 de la version 4.0.0.

6.1.2.1 encodage écran

le test a été effectué avec 2 normes :

ISO-8859-1 UTF8

Attention, fpdf ne supporte pas utf8 et des modifications ont été faites dans la version 4.01 du framewrk pour qu'utf8 soit implémenté (commande utf8_decode et encode)

De la même manière, des bugs ont été corrigés dans la recherche table et dans les sous formulaire (évolution openMairie 4.1.0)

voir framework/paramètrage/locales.inc

6.1.2.2 encodage postgresql

information geographique/install postgis

Il est possible de conserver l'option sql_ascii en création de base avec la version postgresql 8.4 en rajoutant l'option T

```
createdb -E SQL_ASCII -T template0 openelec
```

6.1.2.3 traduction

Il vaut mieux mettre les accents dans les traductions plutôt que de les intégrer dans les zones à traduire et éviter ainsi les problèmes d'encodage.

voir outils/poedit

6.1.3 Tags dans le code PHP

Utilisez 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.

6.1.4 Les normes à respecter

6.1.4.1 Pourquoi respecter des normes ?

...

6.1.4.2 XHTML Valide et le W3C

...

6.1.5 Les commentaires dans le code

Tous les fichiers PHP doivent avoir un entête de ce style

```
<?php
/**
 * Courte description du fichier
 *
 * Description plus détaillée du fichier (si besoin en est)...
 *
 * @package openmairie
 * @version SVN : $Id$
 */

?>
```

6.1.6 Séparation du contenu et de la présentation dans le couple XHTML CSS

...

6.1.7 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.

6.2 Le versionning du code et la version des applications

6.2.1 Convention de numérotation des versions des applications et librairies

Il est convenu de numéroter les versions sur 3 chiffres séparé par des points.

exemple : openMairie 4.0.0

Le premier chiffre représente une version majeure

Le deuxième chiffre est une évolution mineure

Le troisième chiffre est une correction de bug

Les versions beta sont indiqués en fin de numérotation et ne sont jamais maintenues

openmairie_exemple_4.0.0beta

Seule la dernière version opérationnelle est maintenu

Exemple de versionning (complément de la réunion « dev-openMairie » du 13 juin 2012) :

```
4.2.0alpha1     première version non testée
4.2.0beta1      première version testée par le développeur
4.2.0rc1        première version testée en production (1 site)
4.2.0           première version stable généralisable
```

6.2.2 Passage à la version 4.2.0

La version 4.2.0 du framework prend en charge plus de fonctionnalités et donne toutes possibilité de surcharge aux applications

- surcharge des objets généres 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

6.2.2.1 EXTERNALS.txt

vider les 9 repertoires concernés avant de lancer externals

core est le repertoire contenant la librairie openMairie (4.2.0)

om_theme est le theme de l application

appliquer extermnals openmairie /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
```

Ce qui est spécifique à l'application

```
app     scripts spécifiques de l'application sont à mettre en app
        noter dans specific.txt les spécificités
data    scripts sql d'initialisation de la base de données
dyn     paramétrage de l application
gen     générateur des objets de l application
obj     surcharge des objets de l application générés
        surcharge du core "openmairie" om_dbformdyn , om_formulaire (4.2.0), om_
↪application (util)
sql     requête sql en surcharge de gen/sql
tmp     fichiers temporaires de l application (mettre les droits d'écriture pour␣
↪apache)
trs     fichiers uploadés par l application (mettre les droits d'écriture pour apache
```

6.2.2.2 regenerer les tables avec genfull.php

mettre les droits d ecriture a www-data dans gen : $ sudo chmod-R 777 sur les nouvelles tables gerer par www-data remettre les droits d ecriture $ sudo chmod -R 777

6.2.2.3 modifier les paramètres dyn

locales.inc.php (charset) include.inc.php (core)

6.2.2.4 dans obj

ajouter om_table.class.php, om_dbform.class.php, om_formulaire.class.php

6.2.2.5 Evolution om_sig_point vers om_sig_map

om_sig_map est le nouvel outil sig d openMairie

ne concerne que pgsql

executer le script data/pgsql/ver4.2.0.sql

regenerer les 4 nouvelles tables

ajouter les scripts spécifiques dans /obj et /sql/pgsql

6.2.3 Apache Subversion (SVN)

Site officiel du projet SVN

6.2.3.1 Pré-requis

Installer subversion : http://subversion.apache.org/packages.html

6.2.3.2 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/ a 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/ : ...

6.2.3.3 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".

6.2.3.4 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
```

6.2.3.5 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)

6.2.3.6 Keywords

6.2.3.7 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
- ...

6.2.3.8 Tutoriaux

6.2.3.8.1 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

```
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 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>/php
svn propset svn:externals -F ./EXTERNALS.txt .
svn up
svn ci
```

6.2.3.8.2 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 .
#
```

(suite sur la page suivante)

(suite de la page précédente)

```
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>
```

6.2.4 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 règler 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
cela fait branches/trunk/dossiers_souce

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
```

creer - deplacer (autre exemple) - detruire un repertoire sur svn

```
creer un dossier documentation sur svn depuis une copie loacle
svn import documentation svn+ssh://fraynaud@scm.adullact.net/scmrepos/svn/
↪opencimetiere/documentation

renomer = 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 = deplacer 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
```

(suite sur la page suivante)

(suite de la page précédente)

```
delete  detruire 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
```

6.2.5 Concurrent versions system (CVS)

Site officiel du projet CVS

Il est noté dans ce chapitre les commandes de bases de cvs à titre d'indicatif. Les développeurs d openMairie souhaitent qs'orienter sur SVN qui est plus facile dans la gestion des répertoires (notament la suppression) et dans la mise en place de composants externes (externals)

voir outils/svn et outils/svn_utilisation

6.2.5.1 forge > local

update

```
$ cvs up
$ cvs up -A // depuis la branche principale
$ cvs up -r branch_name // depuis une branche
$ cvs up -r tag_name  // depuis un tag
$ cvs up -d  //creer sur le poste local les nouveaux repertoires
$ cvs up -C  //Annulation des modifications locales -> les fichiers remplacés sont␣
↪sauvés en .#file.revision
```

message

```
A Ajout de nouveaux éléments à la version locale
M éléments modifiés localement (par rapport à la version de CVS) ;
? éléments inconnus de CVS (non présents dans la version de CVS) ;
U pour les éléments modifiés dans CVS (par rapport à la version locale) ;
C pour les éléments différents entre les versions locale et CVS, et qui posent un␣
↪conflit à règler manuellement.
D fichier supprimés
```

6.2.5.2 local -> forge

commit

```
$ cd <répertoire désiré>
$ cvs commit [-m "message explicite de la modification"] [<arborescence>]
$ cvs ci -m "ajout des fichiers" // avec message
$ cvs ci module_name // dans la branche principale
$ cvs ci -d branch_name module_name // dans une branche
$ cvs ci -r tag module_name // depuis un tag
$ cvs ci -r 3.0 //changer la révision (doit être plus grand que tous les numéros␣
↪existants)
```

Add

```
$ cd <répertoire désiré>
$ cvs add [-m "message explicite de l'ajout"] <arborescence>
$ cvs add file1 file2 ...
cvs add /data/pgsql*  // * tous les fichiers du repertoire  ...
L'ajout d'un fichier n'est possible que depuis le répertoire le contenant ;
Les fichiers d'un répertoire non ajouté ne sont pas visibles par CVS (lors de l
↪'affichage des différences entre les versions locale et CVS) ;
L'ajout d'éléments est effectif dans CVSROOT uniquement après la mise à jour de la␣
↪version de CVS.
```

rm

```
Pour supprimer des éléments à la version locale, afin de les supprimer ensuite␣
↪définitivement de la version CVS :
$ cd <répertoire désiré>
$ cvs remove [-f] <arborescence>
 * -f : pour effacer le fichier avant de le supprimer.

Il faut d'abord supprimer le fichier du répertoire
$ rm file1 file2 ...
$ cvs rm file1 file2 ...
$ cvs ci -m "suppression des fichiers
Supprimer un répertoire n est pas possible directement, il faut aller le supprimer␣
↪dans le serveur.
```

6.2.5.3 local

Suppression d'une copie locale

```
// pour éviter d'effacer un checkout en oubliant nos modifications
Pour effacer les sources présentes sur le poste local et faire confiance au␣
↪repository :
$ cd <répertoire désiré>
$ cvs release [-d] <arborescence>
* -d : pour effacer définitivement l'arborescence.
$ cvs release -d module
```

status

```
Pour obtenir le détail (statut) de la version locale d'une arborescence :
$ [ cd <répertoire désiré> ]
$ cvs status <arborescence>
cvs st -> status
```

diff/log

```
cvs diff
cvs log <nomfichier>
cvs diff <nomfichier>     [local /differentiel]
cvs diff -r 1.5 -r 1.6 nomfichier [entre 2 versions]
```

6.2.5.4 export

exemple opencimetiere_facture

- se mettre dans le repertoire : openmairie_cimetiere_facture
- taguer la version

```
$ cvs tag version_2_03 (commence par une lettre / pas de point)
```

- faire l export

```
$ cvs export - r version_2_03 -d version/opencimetiere_facture_2.03 openmairie_
↪cimetiere_facture
                          [      version    ]   [repertoire export ] [module]
```

6.2.5.4.1 tag

Les identifiants logiques (noms donnés à une version par un utilisateur) sont différents des identifiants CVS (du type 1.1.2.1). La gestion d'identifiant (ou tag) d'une arborescence se fait ainsi

```
$ [ cd <répertoire désiré> ]
$ cvs tag [-R] [-d -r] <nom de l'identifiant> [<arborescence>]
$ cvs tag tag_name // creer un tag
* -R : commande appliquée récursivement sur les sous-répertoires ;
* -d -r : suppression de l'identifiant existant.
```

Exportation (mêmes options que cvs check out)

Pour exporter les sources du projet en vue d'une livraison (pas de répertoires CVS dans l'arborescence)

```
$ cd <répertoire désiré>
$ cvs export (-r <nom du tag> | -D <date désirée>) <arborescence>

$ cvs export
```

Les fichiers .cvsignore sont exportés et apparaissent dans l'arborescence contrairement aux répertoires CVS ; Des problèmes apparaissent lors d'export de fichiers binaires sur plateformes hétérogènes. Par exemple, un export sur PC transforme les retours chariots (n -> r n)

6.2.5.5 Import

requete cvs

```
cvs -d :pserver:user@cvs.mpl.ird.fr:/projet login
       (1)     (2)      (3)          (4)
        |       |        |            |
        |       |        |            +- Répertoire du
        |       |        |    SERVEUR contenant les sources
        |       |        |              (racine)
```

(suite sur la page suivante)

(suite de la page précédente)

```
        |       |       |
        |       |       |
        |       |       +-------- adresse du SERVEUR CVS
        |       |
        |       +------ Votre login à vous sur le SERVEUR
        |
        +----- le type d'authentification
```

6.2.5.6 checkout

Pour récupérer les sources du projet en local

```
$ cd <répertoire désiré>
$ cvs checkout <arborescence>
```

6.2.5.7 Divers

aide

```
$ cvs -H nomdecommande
```

log des commit : Pour obtenir l'historique d'une arborescence

```
$ [ cd <répertoire désiré> ]
$ cvs log <arborescence>
```

L'historique affiche les différents identifiants (ou tag) et les différentes versions de l'arborescence concernée sous CVS ;

L'historique sur un répertoire affiche récursivement les historiques des fichiers le composant.

.cvsignore

La présence de fichier(s) .cvsignore dans un répertoire permet de dire à CVS d'ignorer certains types de fichiers

```
$ cd <répertoire désiré>
$ cat .cvsignore
*.jpg *.htm
```

6.2.6 Changer le système de gestion des version de CVS vers SVN sur la forge de l'adullact

Le but de ce tutorial est de changer le système de gestion de version sur un projet existant sur la forge de l'adullact.

6.2.6.1 Pré-requis

- Le projet sur l'adullact en CVS <NOM_DU_PROJET>
- Le nom du module à récupérer <NOM_DU_MODULE>
- Les droits d'administration sur le projet <NOM_DU_DEVELOPPEUR>

6.2.6.2 Etape 1 - Récupérer le code du CVS

```
cvs -d :pserver:anonymous@scm.adullact.net:/cvsroot/<NOM_DU_PROJET> login
cvs -d :pserver:anonymous@scm.adullact.net:/cvsroot/<NOM_DU_PROJET> export -DNOW <NOM_
↪DU_MODULE>
```

Important : si un mot de passe est demandé, un mot de passe vide fera l'affaire.

A cette étape, il est recommandé de faire une archive du dossier "<NOM_DU_MODULE>" qui vient d'être exporté pour le sauvegarder.

6.2.6.3 Etape 2 - Changer le type de dépôt

En tant qu'administrateur, aller dans l'onglet "Sources" puis cliquer sur le lien "Administration". Choisir alors SVN puis cliquer sur le bouton "Mettre à jour".

Il faut ensuite attendre (le temps d'attente est variable entre 30 minutes et plusieurs heures) que le dépôt subversion soit activé.

6.2.6.4 Etape 3 - Créer la structure du dépôt

Ici nous allons créer la structure standard d'un dépôt Subversion :

- trunk
- tags
- branches

```
svn mkdir svn+ssh://<NOM_DU_DEVELOPPEUR>@scm.adullact.net/svnroot/<NOM_DU_PROJET>/
↪trunk svn+ssh://<NOM_DU_DEVELOPPEUR>@scm.adullact.net/svnroot/<NOM_DU_PROJET>/tags␣
↪svn+ssh://<NOM_DU_DEVELOPPEUR>@scm.adullact.net/svnroot/<NOM_DU_PROJET>/branches -m
↪"Création de la structure du dépôt Subversion"
```

6.2.6.5 Etape 4 - Importer le code sur le nouveau dépôt Subversion

6.2.6.5.1 Cas 1

Si l'application doit être utilisée telle qu'elle a été récupérée sur le CVS, alors nous allons l'importer directement dans le dossier "trunk".

```
svn import <NOM_DU_MODULE> svn+ssh://<NOM_DU_DEVELOPPEUR>@scm.adullact.net/svnroot/
↪<NOM_DU_PROJET>/trunk -m "Import de la version de l'application anciennement sous␣
↪CVS"
```

6.2.6.5.2 Cas 2

Dans le cas de figure où l'application va être migrée vers OM4, nous allons placer le code récupéré sur le CVS dans une branche du dépôt correspondant à sa version.

```
svn import <NOM_DU_MODULE> svn+ssh://<NOM_DU_DEVELOPPEUR>@scm.adullact.net/svnroot/
↪<NOM_DU_PROJET>/branches/1.x -m "Import de la version de l'application anciennement␣
↪sous CVS"
```

Dans ce cas, il faut donc importer le nouveau code dans le dossier "trunk" :

— Soit le développement du projet n'a pas encore commencé et il suffit de suivre le tutorial « versions/svn/Importer un nouveau projet ».
— Soit le développement du projet a déjà commencé et il suffit d'importer le dossier en cours de développement (attention : il ne faut pas que des dossiers .svn soient présents dans ce dossier et il faut prendre soin de supprimer les dossiers récupérés depuis les EXTERNALS avant l'import) :

```
svn import <NOM_DU_DOSSIER> svn+ssh://<NOM_DU_DEVELOPPEUR>@scm.adullact.net/
↪svnroot/<NOM_DU_PROJET>/trunk -m "Import initial de l'application"
```

Il faut bien sur valider les EXTERNALS pour récupérer les librairies externes.

6.3 Les outils du développeur

Les sites de référence :
— http://php.net/
Les outils indispensables :

6.3.1 Le navigateur Mozilla Firefox

Mozilla Firefox

Ses modules complémentaires indispensables au développement Web :

6.3.1.1 Web Developer

Web Developer

6.3.1.2 Firebug

Firebug

6.3.1.3 HTML Validator

HTML Validator

6.3.1.4 YSlow

YSlow

6.3.2 Komodo Edit

Free Code Editor for Windows, Mac and Linux

http://www.activestate.com/komodo-edit

http://www.activestate.com/komodo-edit/downloads

6.3.3 Meld

exemple d utilisation

```
meld openmairie_recensement/ svn.openmairtrunck/trunk
```

6.3.4 POEdit

...
- TortoiseSVN
- TortoiseCVS

L'environnement de développement :
- Apache
- PHP
- EasyPHP
- Wamp
- Lamp

CHAPITRE 7

Contributeurs

— Florent Michon [ATREAL]
— Francois Raynaud