openMairie - Guide du développeur Documentation

Version 0.0

openMairie

30 December 2010

Table des matières

Ce document a pour but de guider les développeurs dans la mise en oeuvre 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. Bonne lecture, et n'hésitez pas à nous faire part de vos remarques à l'adresse suivante : contact@openmairie.org

Si vous débutez, il est préférable de commencer par le chapître "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 chapître 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 chapître consacré au "générateur" décrit dans le détail le fonctionnement de cet outil et de ses assistants.

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.

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
- Creer 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

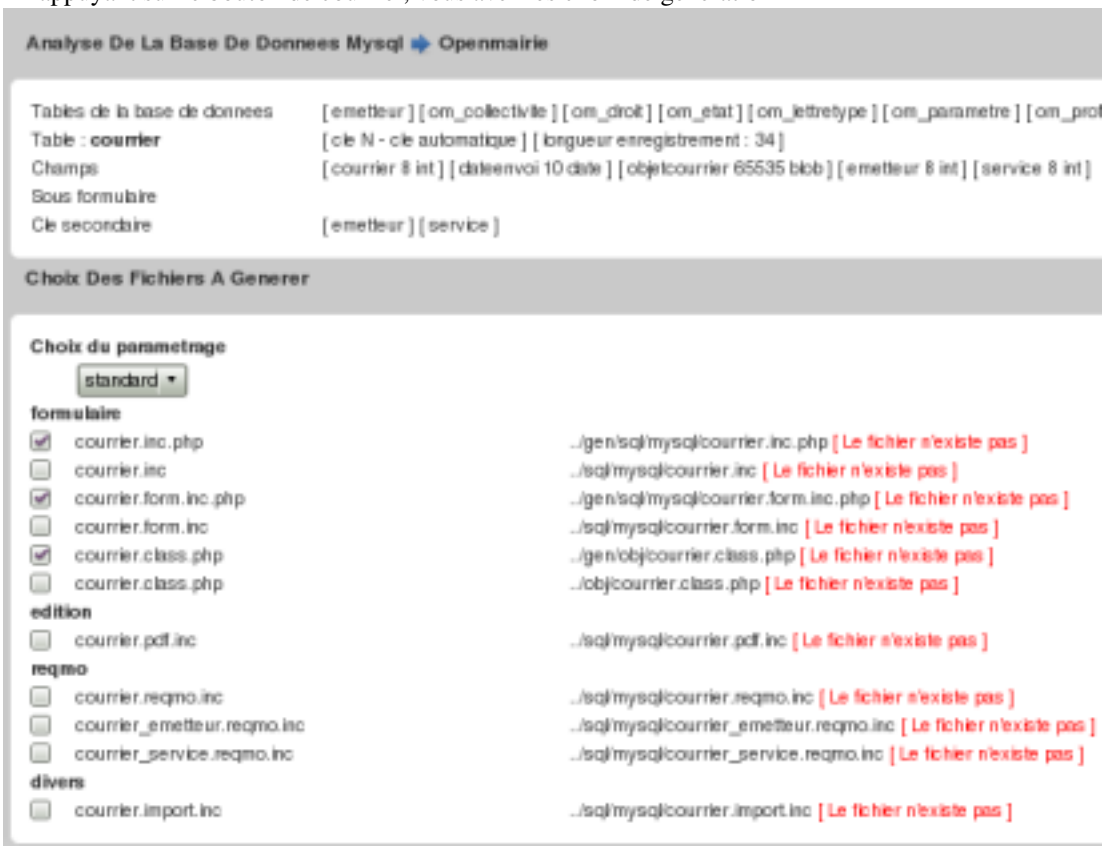
Pour cela, il faut aller dans le menu administration -> generateur

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

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

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 ..

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"
           ));



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

     array_push($links,
       array(
```

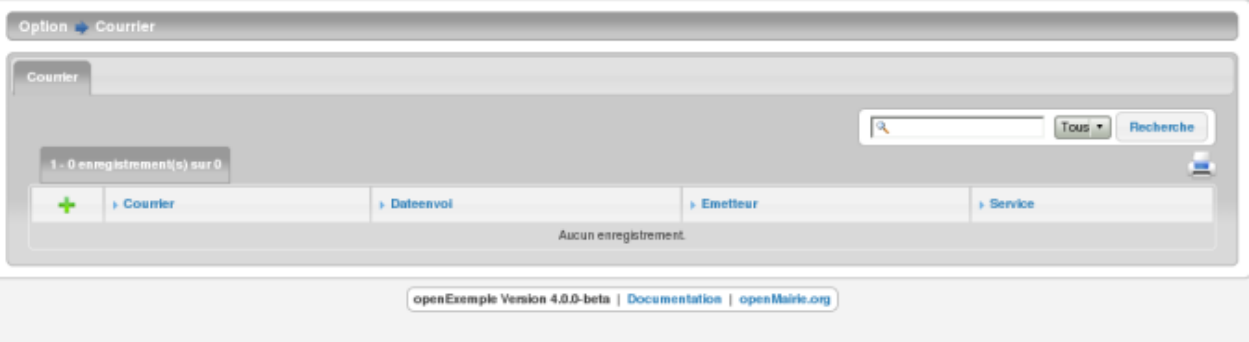
```
        "href" => "../scr/tab.php?obj=emetteur",
        "class" => "emetteur",
        "title" => _("emetteur"),
        "right" => "emetteur"
    ));

    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

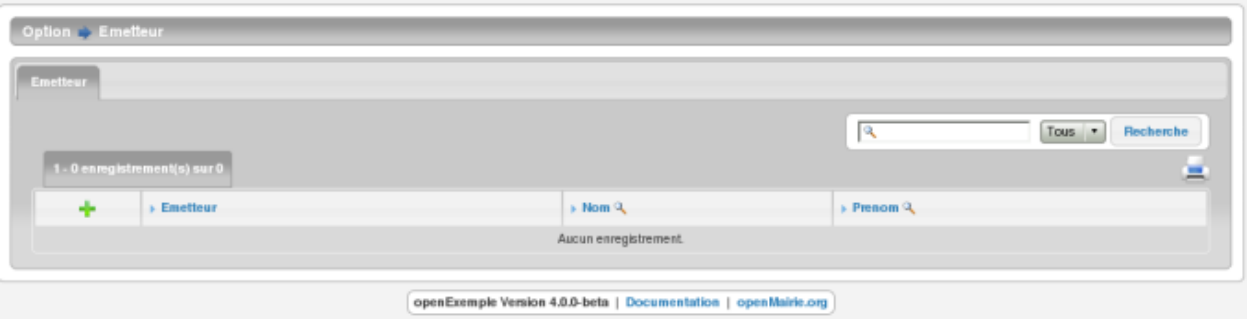
Cette opération affiche la table courrier :



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 :

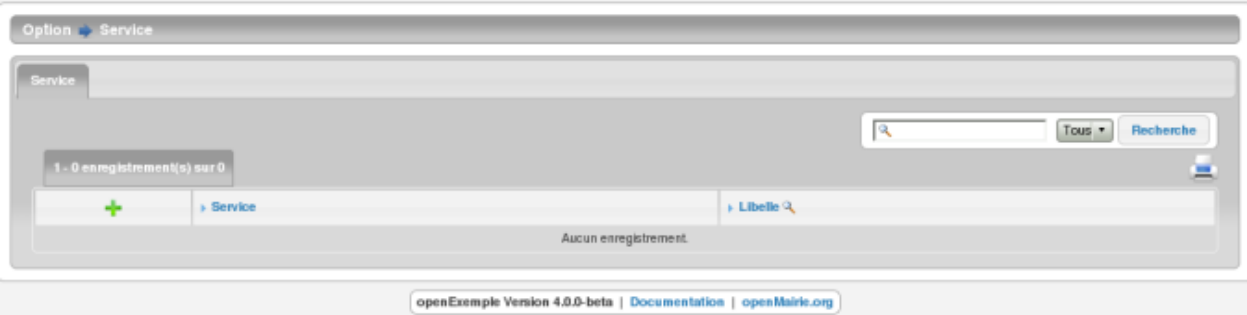


En appuyant sur +, on accède à la saisie

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

parametrage -> service

Cette opération affiche la table service :



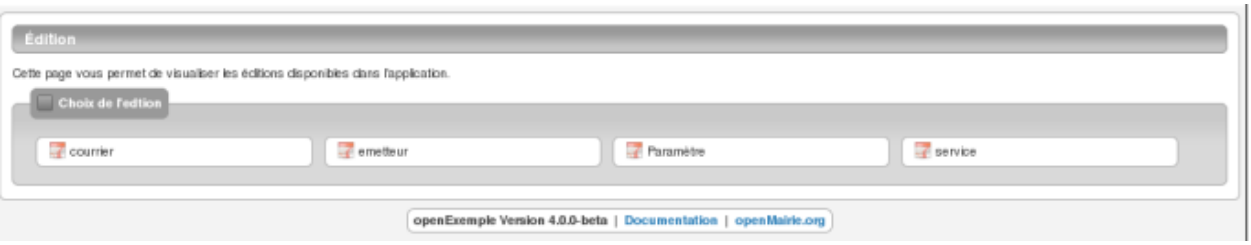
En appuyant sur +, on accede à la saisie

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

Vous pouvez accéder aux éditions et requêtes mémorisées :

export -> edition

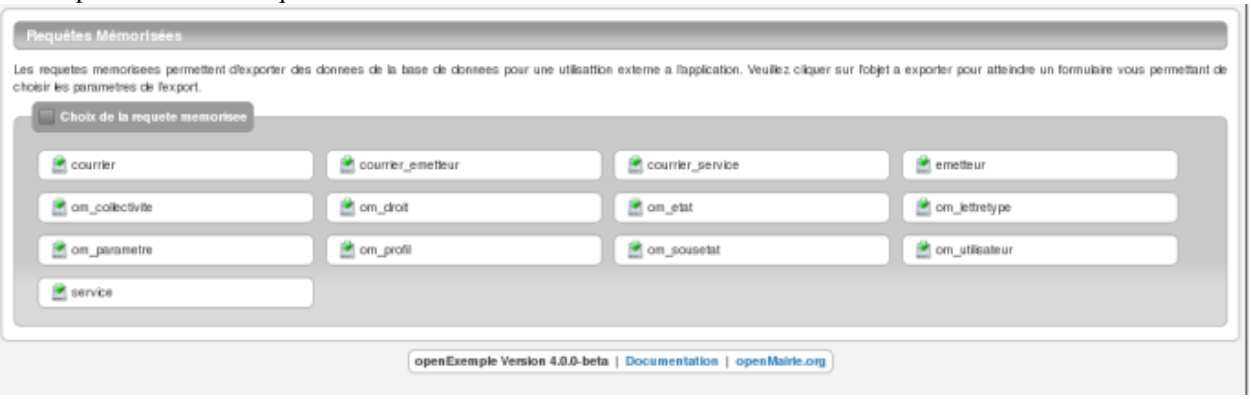
Cet option affiche l'ensemble des éditions pdf :



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'
--

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
                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=" !    ";
    $imgv="<img src='../img/punaise.png' style='vertical-align:middle' hspace='2' border='0'>";
    if ($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 chapître 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)) a
    'concat(emetteur.nom,\' \',emetteur.prenom) as emetteur',
    '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 a 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 chapître *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 chapître 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 :

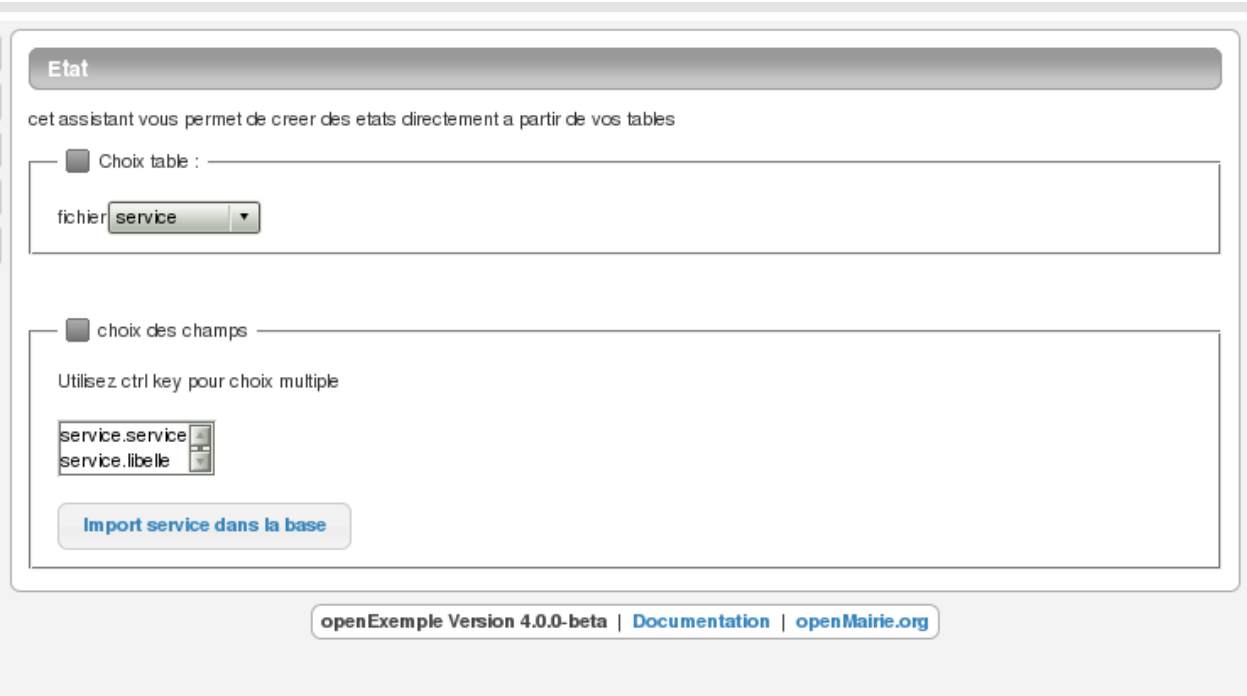
administration -> générateur : assistant

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".

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 :

OpenMairie

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 chapître *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.

Dans un environnement LAMP/WAMP, le framework 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), NSOAP pour les web services, 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 dbformdyn.class.php (composant openMairie). De base, les données de la base de données sont récupérés pour le formulaire (longueur, max, nom).
– dbformdyn.class.php ; assure la liaison entre le formulaire et la base de données
– formulairedyn.class.php : rassemble toutes les méthodes permettant de construire des formulaires

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

    - le paramétrage général du framework
    - la gestion des accès du framework et la multi collectivite
    - les méthodes pour construire des formulaires avec le framework
    - les outils d'édition du framework
    - l'outil de requête paramétrable du framework
    - l'ergonomie intégrant jquery
    - la gestion de traitement et la construction de programme spécifiques avec les utilitaires
    - l'import des données CSV du framework
    - l'interface geographique intégrant openlayers avec openCimetiere
    - l'envoi de mail intégrant phpmail avec openPersonnalite (en projet)
    - le fonctionnement d'artichow dans openResultat (en projet)
    - le fonctionnement du nusoap dans openFoncier (enprojet)
    - l'utilisation de fpdf pour écrire dans un pdf dans openCourrier (en projet)
    - la mise en oeuvre d'un éditeur wysiwyg pour l'édition des états (en projet)
```

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

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

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 (version 4.0.0)

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
administration          Les scripts de cet option contiennent tout les scripts
                            du framework pour le paramètrage de la collectivité,
                            des états / sous états  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/tdb.inc*.

Le parametrage est libre et depend de l'application.

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

Nous proposons cet exemple de code

```
$description = _("Bienvenue ").$_SESSION["login"]."<br>";
$f->displayDescription($description);
```

Ce paramétrage va afficher "bienvenue demo" dans la page d'accueil ou tableau de bord pour l'utilisateur "demo"

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

```
define('CHARSET', 'ISO-8859-1');
```

– 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 chapître 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 - les différents choix possibles se trouvent dans le dossier : ../lib/jquery-ui/css/
Par defaut, le thème d'openExemple est "om_overcast"

```
$config['theme'] = "om_overcast";
```

Les thèmes openmairie_exemple sont : "om_overcast" ; "om_sunny" ; "om_ui-darkness" ;

Vous pouvez mettre d'autres themes jquery.

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

```
define("PATH_OPENMAIRIE", getcwd()."/../php/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

VERBOSE_MODE : mode "bavard"
DEBUG_MODE : mode debug
PRODUCTION_MODE : mode de production (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

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 la version openMairie 4.0.1

2.1.12 Les paramétres des combos

Les paramétres combos sont paramétrés dans les fichiers suivants

- `comboaffichage.inc.php`
- `comboparametre.inc.php`
- `comboretour.inc.php`

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

2.1.13 Les paramétres éditions

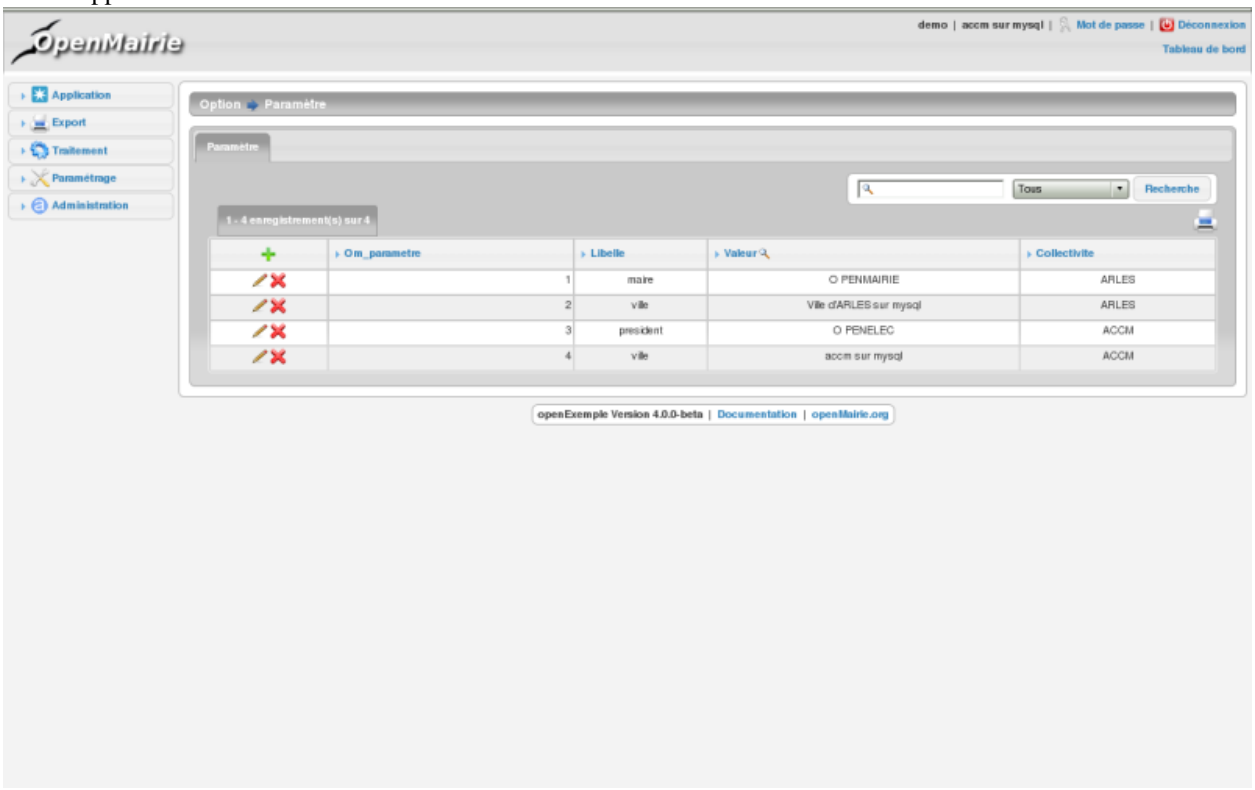
Les variables dans les éditions sont paramétrées dans

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

Voir *chapître framework/édition*

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

$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

$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

```
php/openmairie/om_table.class.php
```

2.3 Les formulaires

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

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

2.3.1 Les methodes de formulairedyn.class.php

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

Les méthodes sur les controles du formulaire

```
Hiddenstatic -> Champ non modifiable  Valeur récupéré par le formulaire
Hiddenstaticdate -> date non modifiable Valeur récupéré par le formulaire
statiq -> Valeur non modifiable
date -> date modifiable  js/calendrier
select -> Controle select
selectdisabled -> Controle select non modifiable
Text -> Controle text
hidden -> Controle non visible avec valeur conservée
password -> Controle password
Textdisabled -> Controle text non modifiable
comboG -> Appel à un programme de correspondance à une table
           Cas ou il y a une grosse table en correspondance
           spg/combo
ComboD -> Appel à un programme de correspondance à une table
           Cas ou il y a une grosse table en correspondance
           spg/combo
Upload -> spg/upload et spg/voir.php
textarea
textarea_html
localisation -> spg/localisation.php
rvb -> spg/rvb.php
```

Les méthodes de construction et d affichage

```
afficher() affichage des champs (appelle par 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)
```

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 php/openmairie/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

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 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

```
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 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 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


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

Exemple avec concession d'openCimetiere

```
dbformdyn.class.php  <- gen/obj/emplacement.class.php <-/obj/emplacement.class.php <- /obj/concessior
```

2.4.2 Modifier les valeurs par défaut

Il est décrit ici les valeurs par défaut dans php/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 dbformdyn.class.php

dbform.class.php est une classe openMairie

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 donn
setGroupe : Envoi au formulaire le regroupement de champ par ligne
setRegroupe : Envoi au formulaire un fieldset
```

– 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] =>
                    [_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] => )
    [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, le titre et la requete sql, 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 et la requete sql, 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, le titre et la requete sql, 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 et la requete sql, 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 la version openMairie 4.0.1..

2.5.7 L'éditeur WYSIWYG

Un editeur est prevu dans la prochaine version openMairie 4.0.1

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 la version 4.0.1 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

Les composants php sont stockés en **php/**

/openmairie

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

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

/fpdf

A ce niveau se situe le composant fpdf

/phpmailer

la gestion de mail est EN TEST avec openPersonnalite et sera intégré dans openMairie 4.0.1

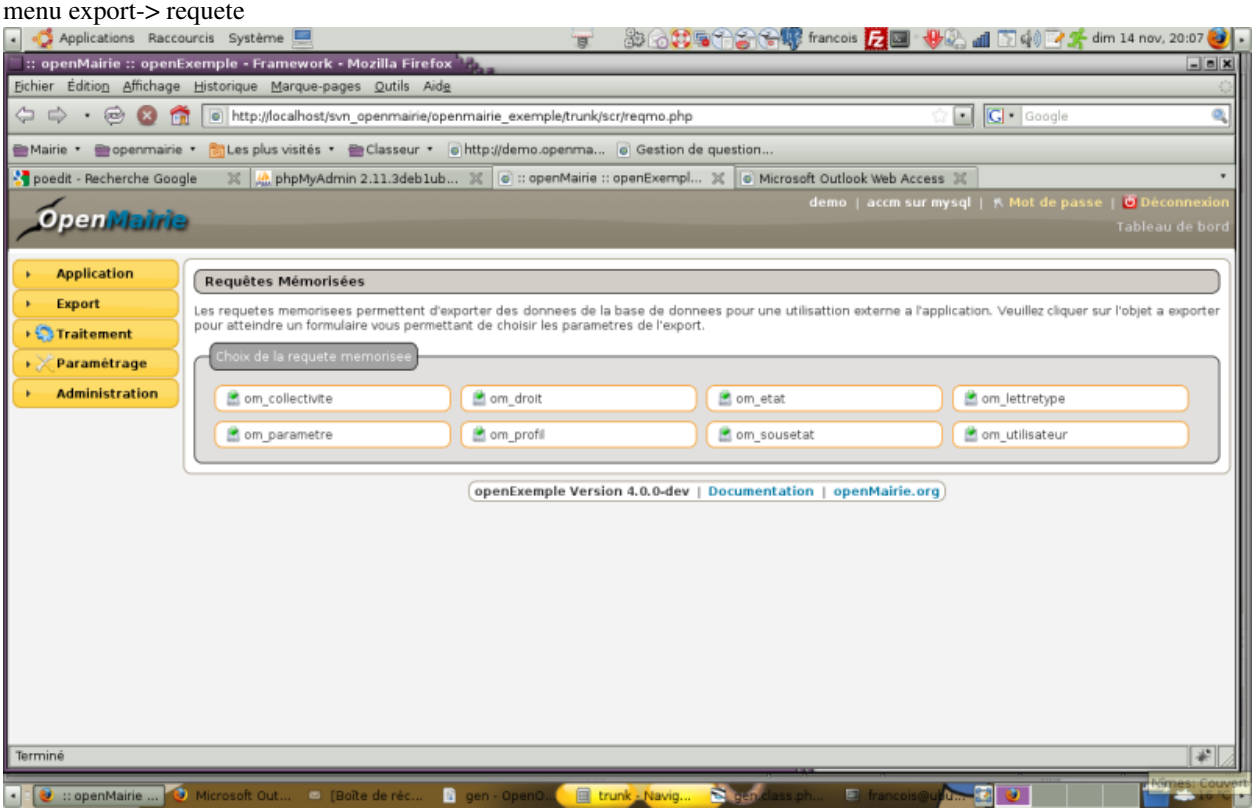
Les composants javascript sont stockés dans le repertoire **lib/**

/tinymce

est l'editeur wisiwig EN TEST sur openRecensement et qui sera intégré dans openmairie 4.0.1)

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 choisi 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 chapître 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

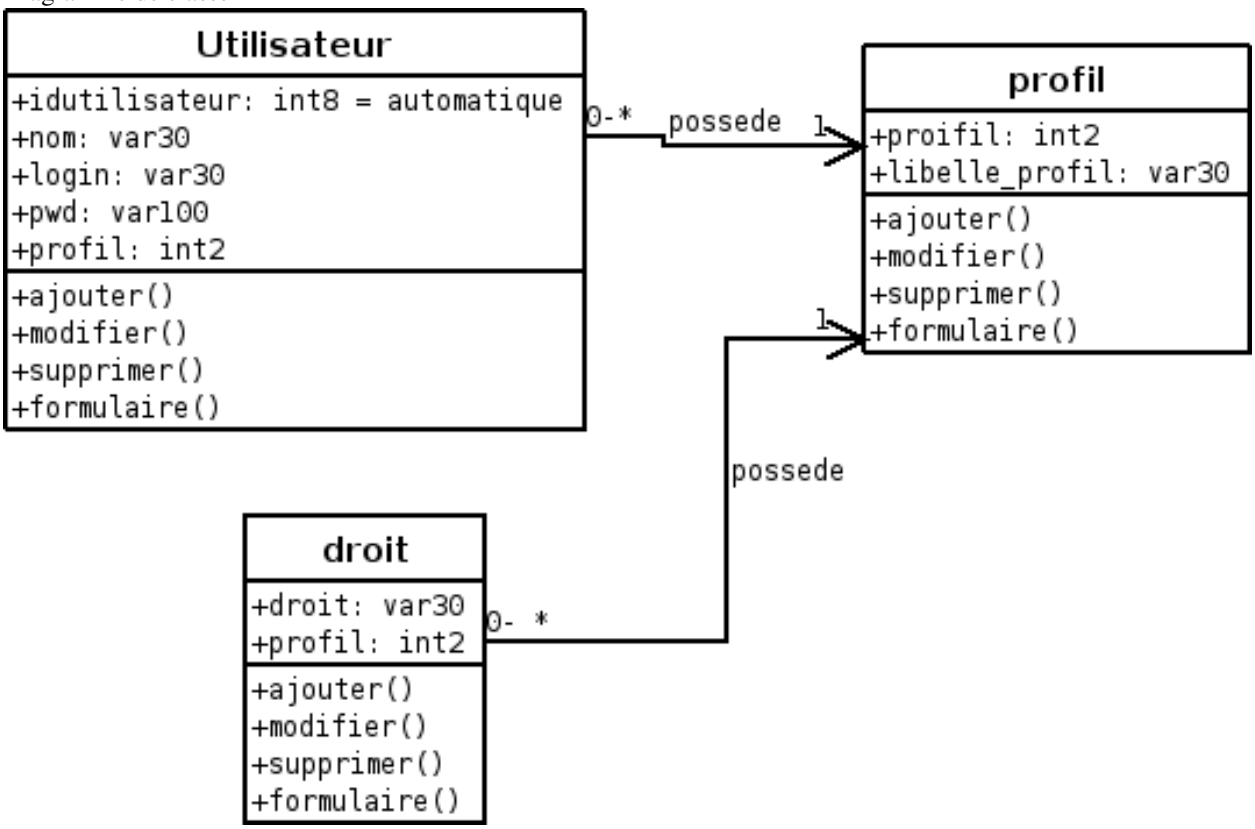
```
administrateur
super utilisateur
utilisateur
utilisateur limite
consultation
```

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
- chaque profil a acces a tous les droits des profils d un niveau inférieur
- l'adminitrateur 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 décidé d'utiliser l'ergonomie de jquery.

2.8.1 Le composant jquery

Les skins jquery peuvent être rajoutés dans le repertoire /lib/jquery_ui.

Le changement de skin se fait dans le fichier dyn/config.inc.php

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
    specific.css : specifique a l application
        specific_om_... : suivant la feuille de style jquery choisi dans l application (1)
```

2.8.3 La mise en oeuvre dans les scripts

voir framework/utilitaire

2.9 Les utilitaires

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 peuvent être créer pour :
– faire un traitement en répertoire trt/ (remise à 0 d'un registre, archivage, export ....)
– faire un sous programme spécifique en spg/ appellé par un formulaire (bible.php dans openCourrier)
– faire une recherche avec un affichage particulier en scr/.

2.9.1 Le tutorial

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

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 défqut *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 11:43:36Z xxxxx $
 */


// CREATION DE L' OBJET $f

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



// 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\">";
    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
```

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"

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
```

2.11 Intégrer le système information geographique

La plupart des applications openMairie sont géo localisables

Il a été choisi de géolocaliser les données en utilisant :
– postgresql et sa cartouche graphique postgis
– openlayers pour afficher et mettre à jour des données graphiques
Ce module est opérationnel avec la version 2.02 d'openCimetiere et openLayer est implémenté dans openMairie_exemple 4.0.0.

Pour l'instant les scripts d'intégration ne sont pas mis en place. Il le seront dans la version 4.0.1 dans le répertoire /sig.

2.11.1 Les scripts sig

Dans le répertoire sig, il y a 3 scripts :
– tab_sig.php : affiche la carte suivant les parametres de nom_objet.sig.inc
– form_sig.php : modifie le champ geometry de l'objet
– delete_sig.php : supprime le contenu du champ geometry
Il y a un repertoire map ou sont stockés les fichiers map utilisés par mapserver

Le fichier var.inc stocke les variables suivantes

```
// chemin de la carte pour mapserver
$path_map="http://localhost/cgi-bin/mapserv?map=/var/www/
                  openmairie_cimetiere/sig/map/";

// polygon créé par défaut
$creation_polygon="POLYGON((0 0,0 0))";
```

Le fichier style.css contient la css de l interface

Le lancement du sig se fait avec tab_sig.php sous la forme suivante :

/sig/tab_sig ?idx=29&obj=emplacement

où :

$idx est l' enregistrement ou modifier/ajouter/supprimer la geometry dans le champ geom

$obj est la table ou la geometry doit etre modifiée : cimetiere, emplacement, voie, zone

ATTENTION, il ne peut etre saisi qu un enregistrement existant dans la table

Dans pgsql, il y a les parametres des affichages tab_sig.php

En effet, il s agit d'afficher pour chaque objet la partie (box) de la carte à afficher et l'objet courant et le titre

exemple : objet concession :

L'objectif dans openCimetiere est de saisir d'un polygone geometrique representant un emplacement en fonction de la zone

La procedure est la suivante :

-> afficher une partie de la carte du cimetière (la zone de l'emplacement)

-> dessiner un polygone représentant la géometrie de la concession saisie

-> enregistrer la geometry (wkt) : form_sig.php ou supprimer : delete_sig

les paramétres sig sont dans sql/pgsql/emplacement.sig.inc :

```
$fichier_map ="cimetiere.map";                          nom de la carte dans /map
$titre= "Emplacement ".$idx." - ";                      titre affiché
$texte="";                                              texte affiché
$aff="zone";                                            fond de carte
$class_map="bigmap";                                    taille : smallmap pu bigmap
$sql_idgeom="select voie.zone from emplacement " .      Identifiant a afficher
    "inner join voie on emplacement.voie=voie.voie " .
        "inner join zone on zone.zone=voie.zone
        where emplacement.emplacement =".$idx;
// box et zone
$affiche='famille';                                     champ affiché sur  emplacement
$sql ="select ('cimetiere ' ||cimetierelib||'       selection du libelle et de la box
        zone '|| zonelib) as lib,box(zone.geom)
        as box  from zone inner join cimetiere
        on cimetiere.cimetiere=zone.cimetiere ";
```

2.11.2 Installation

Il est necessaire d(installer sur le serveur apache, mapserver.

Il faut aussi installer la cartouche postgis avec postgres

Les installations sous UBUNTU sont les suivantes

```
* installer les paquets postgis
- postgis
- postgresql-8.3.postgis

* installer mapserver
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
(optionnel)
- php5-mapscript.
relancer apache  $  /etc/init.d/apache2 restart

*** Installation de la base opencimetiere avec postgis

* creer la base opencimetiere (si elle n'est pas deja creee)
* 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 opencimetiere
    * data/pgsql/init.sql
    * data/pgsql/initsig.sql
    * data/pgsql/initsig_data.sql (optionnel) jeu de donnees
```

CHAPITRE 3

Le générateur

3.1 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

3.1.1 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 multicolle
```

La version 4.00 ne reprend pas (par rapport a la version 3) :

```
l'état reprenant l'enregistrement et les sous états rattachés  à l'état principal par la ou les clés

la documentation avec un lien sur les pages des tables reliées par la clé secondaire et l'option dans

l'option dans le menu et le tableau de bord

la prise en compte dans la recherche globale
```

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.

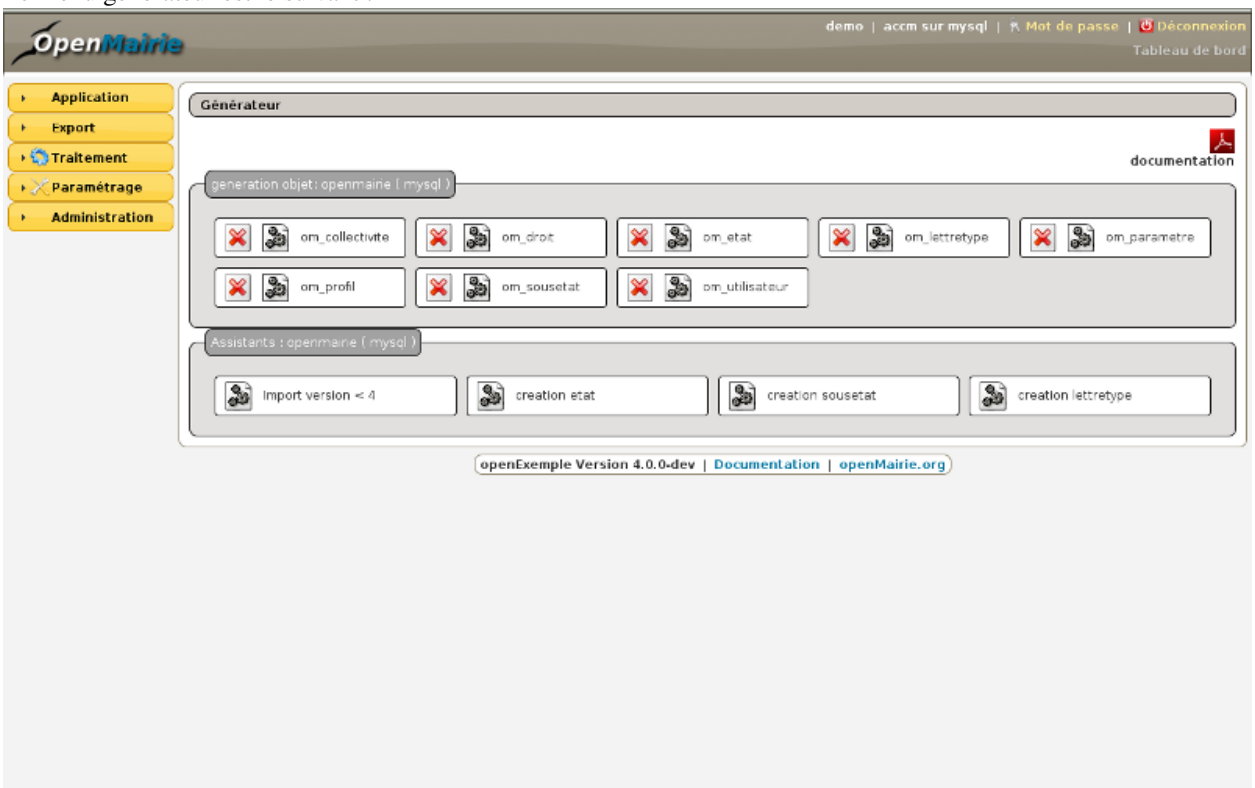
openMairie est multi collectivité : les états et les sous états sont générés dans la base de données et peuvent être associé a une collectivité.

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

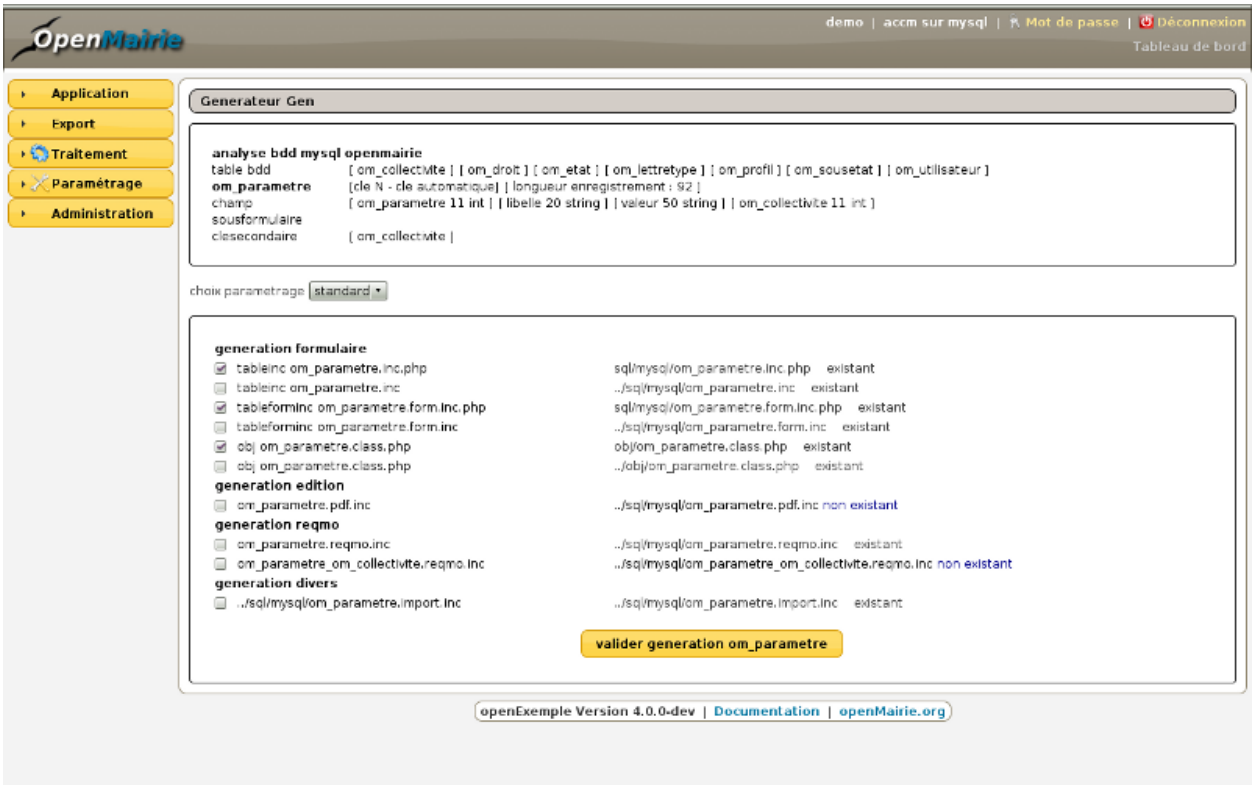
Les schemas et prefixes sont gérés

3.1.2 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
le choix du fichier de paramétrage par défaut



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

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
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

3.1.3 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 4 types de champs reconnus par le générateur :

```
- string : chaîne de caractère
- int : nombre (entier ou décimal)
- date
- blob : texte
```

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
```

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
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
```

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_
```

Gestion des longueurs négatives :

Il est remonté dans l'analyse des informations de la base une valeur négative car le stockage dans postgresql est variable. OpenMairie utilise alors la longueur d'affichage qui est donnée par la plus longue saisie dans un champ. Donc pour avoir la longueur exacte, il est conseillé de saisie un enregistrement avec les champs «string» saisie avec le nombre de caractères souhaités à l'affichage.

Dans le cas ou rien n'est saisi, il est proposé dans form.inc

```
$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
```

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

3.1.4 Les fichiers générés

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

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.

Paramétres de type table :

– gen/sql/base/nom_table.inc
– sql/base/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

Paramétres de type Form :

gen/sql/base/nom_table.form.inc

sql/base/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

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
. 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

La longueur d'affichage et le maximum autorisé à la saisie est celle contenu dans la base d'origine (retraitée pour les champs string de pgsql)

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

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

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
– 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)

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.

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.

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.

3.1.5 Paramétrage générateur

Le paramétrage de base est dans le répertoire gen/dyn/standard. Il est possible de le modifier dans le répertoire custom ou dans un autre répertoire. Le choix du paramétrage se fait à chaque génération.

le paramétrage standard

Le paramétrage standard est dans le répertoire gen/dyn/standard

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_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
$pgsql_longueur_date=12;              taille d'affichage de la date '
```

Etat.inc

Parametres

```
$variable='&'; (indice precedent la variable)
// general
$orientation='P';
$format='A4';
// footer
$footerfont='helvetica';
$footerattribut='I';
$footertaille='8';
// logo
$logo='logopdf.png';
$logoleft='58';
$logotop='7';
// titre
$titreleft='41';
$titretop='36';
$titrelargeur='130';
$titrehauteur='10';
$titrefont='helvetica';
$titreattribut='B';
$titretaille='15';
$titrebordure='0';
$titrealign='C';
// corps
$corpsleft='7';
$corpstop='57';
$corpslargeur='195';
$corpshauteur='5';
$corpsfont='helvetica';
$corpsattribut='';
$corpstaille='10';
$corpsbordure='0';
$corpsalign='J';
// sous etat
$se_font='helvetica';
$se_margeleft='8';
$se_margetop='5';
$se_margeright='5';
$se_couleurtexte="array('0','0','0')";
```

Sousetat.inc

Parametres

```
$longueurtableau= 195;
$variable='&'; (indice precedent la variable)
//titre
$titrehauteur=10;
$titrefont='helvetica';
$titreattribut='B';
$titretaille=10;
$titrebordure=0;
$titrealign='L';
$titrefond=0;
$titrefondcouleur="array(255,255,255)";
$titretextecouleur="array(0,0,0)";
// intervalle
$intervalle_debut=0;
$intervalle_fin=5;
// entete
$entete_flag=1;
$entete_fond=1;
$entete_orientation="array(0,0,0)";
$entete_hauteur=7;
$entete_fondcouleur="array(255,255,255)";
$entete_textecouleur="array(0,0,0)";
// tableau
$tableau_bordure=1;
$tableau_fontaille=10;
// bordure
$bordure_couleur="array(0,0,0)";
// sous etat fond
$se_fond1="array(243,246,246)";
$se_fond2="array(255,255,255)";
// cellule
$cellule_fond=1;
$cellule_hauteur=7;
// total
$cellule_fond_total=1;
$cellule_fontaille_total=10;
$cellule_hauteur_total=15;
$cellule_fondcouleur_total="array(255,255,255)";
// moyenne
$cellule_fond_moyenne=1;
$cellule_fontaille_moyenne=10;
$cellule_hauteur_moyenne=5;
$cellule_fondcouleur_moyenne="array(212,219,220)";
// nombre d enregistrement
$cellule_fond_nbr=1;
$cellule_fontaille_nbr=10;
$cellule_hauteur_nbr=7;
$cellule_fondcouleur_nbr="array(255,255,255)";
```

Pdf.inc

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";
$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";
$C1entetetxt=0;// couleur texte R";
$C2entetetxt=0;// couleur texte V";
$C3entetetxt=0;// couleur texte B";
$C1border=159;// couleur texte  R";
$C2border=160;// couleur texte  V";
$C3border=167;// couleur texte  B";
$bt=1;// border 1ere  et derniere ligne  du tableau par page->0 ou 1";
```

Customiser le paramétrage

Il est possible de personnaliser le paramétrage dans le répertoire custom ou en créant un autre répertoire avec des paramètres personnels.

Il faut mettre dans le répertoire le où les fichiers à personnaliser.

Ne personnaliser que les variables souhaitées dans le fichier. Par défaut, openMairie prendra les paramètres standards.

Choisir le paramétrage personnalisé dans l'écran de génération qui affiche les répertoires de paramètres existants.

Ne pas supprimer le répertoire standard et les fichiers par défaut.

CHAPITRE 4

Règles et outils

4.1 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.

4.1.1 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.

4.1.2 L'encodage des fichiers

Il vaut mieux mettre les accents dans les traductions plutôt que de les intégrer dans les zones à traduire.

voir poedit

4.1.3 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.

4.1.4 Les normes à respecter

Pourquoi respecter des normes ?

...

XHTML Valide et le W3C

...

4.1.5 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 : $Id$
 */

?>
```

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

...

4.1.7 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.

4.2 Le versionning du code et la version des applications

4.2.1 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 deuxièeme chiffre est une évolution mineure

Le troisième chiffre est une correction de bug

Les versions beta sont indiqué en fin de numérotation et ne sont jamais maintenu.

Seule la dernière version opérationnelle est maintenu

4.2.2 Apache Subversion (SVN)

Site officiel du projet SVN

Pré-requis

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

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/ : ...

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'.

Les commandes basiques à connaître

Récupérer une copie locale :

```
svn co svn+ssh://nom-du-développeur@scm.adullact.net/openmairie/openmairie_exemple/trunk openmairie_e
```

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
```

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 .
#

openmairie svn://scm.adullact.net/svnroot/openmairie/openmairie/tags/4.0.0/
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/
```

Appliquer les propriétés externals :

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

Keywords

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
– ...

Tutoriaux

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/<DERN
```

On cré l'arborescence standard sur le dépôt

```
svn mkdir svn+ssh://<NOM_DU_DEVELOPPEUR>@scm.adullact.net/svnroot/<NOUVEAU_PROJET>/trunk
svn mkdir svn+ssh://<NOM_DU_DEVELOPPEUR>@scm.adullact.net/svnroot/<NOUVEAU_PROJET>/tags
svn mkdir svn+ssh://<NOM_DU_DEVELOPPEUR>@scm.adullact.net/svnroot/<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/svnroot/<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
```

Publier une nouvelle version

Ce tutorial contient certains pré-requis comme le fait d'avoir un utilisateur avec les droits corrects sur le projet ou connaître comment incrémenter le numéro de version de l'application à publier.

Avant de publier une application, il faut vérifier que l'EXTERNALS de la librairie openMairie ne pointe pas vers le 'trunk'. Pour cela

```
less php/EXTERNALS.txt

#
# created by: svn propset svn:externals -F ./EXTERNALS.txt .
#

openmairie svn://scm.adullact.net/svnroot/openmairie/openmairie/trunk/
fpdf svn://scm.adullact.net/svnroot/openmairie/externals/fpdf/tags/1.6-min/
pear http://svn.php.net/repository/pear/pear-core/tags/PEAR-1.9.1/
db http://svn.php.net/repository/pear/packages/DB/tags/RELEASE_1_7_13/
```

Ici on voit que openmairie pointe vers le 'trunk'. Nous devons d'abord publier la librairie

```
svn cp svn+ssh://<NOM_DU_DEVELOPPEUR>@scm.adullact.net/openmairie/openmairie/trunk svn+ssh://<NOM_DU_
```

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:/
```

4.2.3 Concurrent versions system (CVS)

Site officiel du projet CVS

4.3 Les outils du développeur

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

4.3.1 Le navigateur Mozilla Firefox

Mozilla Firefox

Ses modules complémentaires indispensables au développement Web :

Web Developer

Web Developer

Firebug

Firebug

HTML Validator

HTML Validator

YSlow

YSlow

4.3.2 Komodo Edit

Free Code Editor for Windows, Mac and Linux

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

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

4.3.3 Meld

4.3.4 POEdit

...
– TortoiseSVN
– TortoiseCVS
L'environnement de développement :
– Apache
– PHP
– EasyPHP
– Wamp
– Lamp

CHAPITRE 5

Contributeurs

- Florent Michon [ATREAL]
- Francois Raynaud