LABINVENT DOCUMENTATION

URL officielle de ce doc : https://tinyurl.com/labinvent2

Authors: E. Pallier, E. Bourrec

⇒ **Pour faire une copie** Word ou Open Office ou PDF de ce doc : menu “Fichier/Télécharger…”

TABLE DES MATIÈRES

1. INSTALLATION

(updated 13/12/18)

Cette doc est en cours de migration à l'intérieur du présent document.

En attendant, voici le lien vers la doc d'origine

2. WORKFLOW GENERAL

2.1. Description générale

(updated 12/12/18)

Les materiels figurent dans Labinvent s'ils font plus de 800 E, ou s'il y a un interet technique a inventorier ce type de matériel.

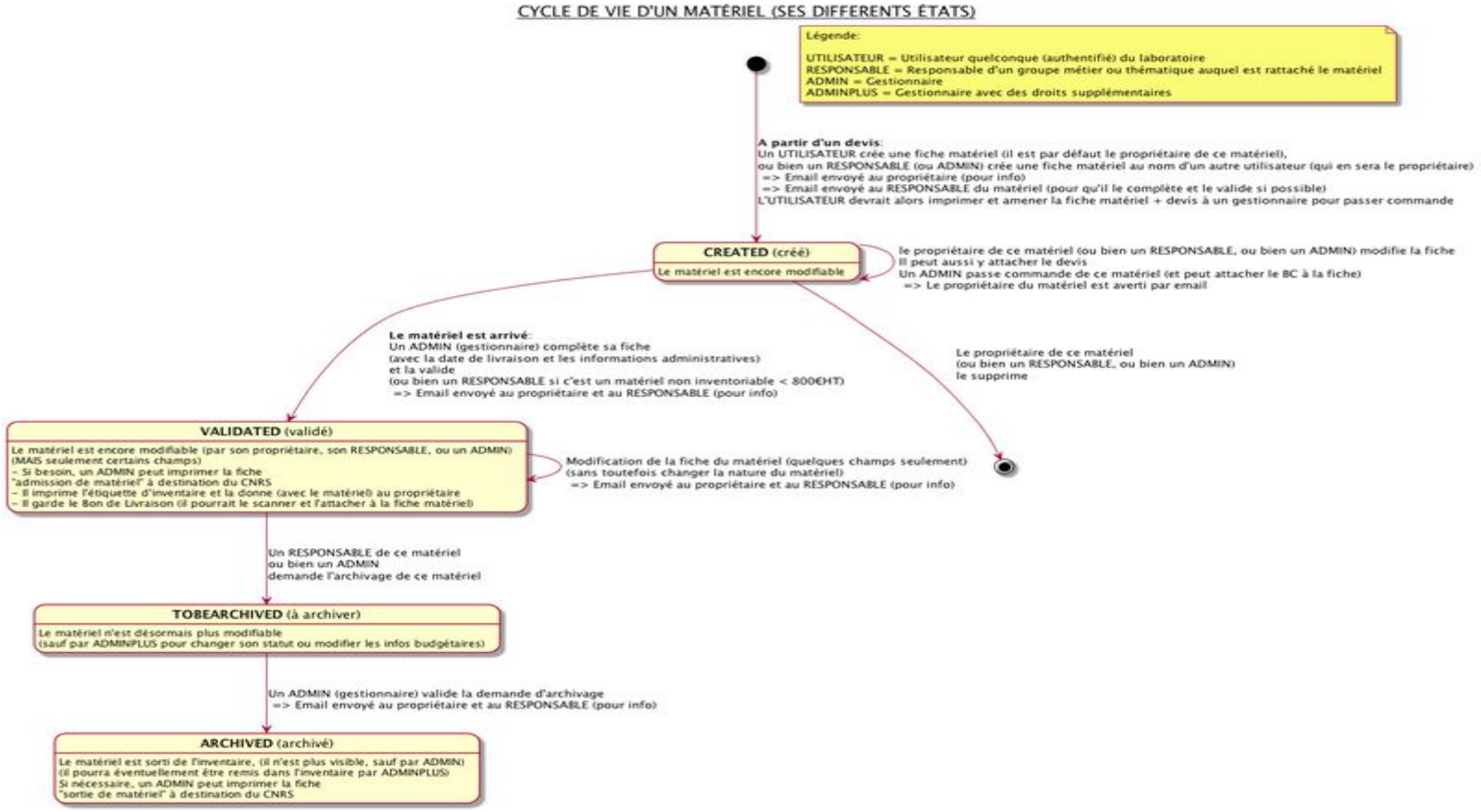
Un usager veut commander un matériel :
Il va creer une fiche (avec toutes les caracteristiques et description appropriee) et avoir un munéro d'inventaire. Le matériel a un statut "Created". Il peut eventuellement etre supprimé.
Avec ce numéro (ou la fiche) il envoie sa demande (avec le devis) a son personnel de gestion qui va emettre le bon de commande, et eventuellement, sur Labinvent, remplir la partie administrative du matériel en question.

Qd le matériel arrive :
Avec le bon de livraison, le matériel peut etre validé (c'est la que certaines questions d'organisation, et de droit sur l'appli, se posent) Les gestionnaires peuvent valider les materiels mais souvent, ils n'en voient que le Bon de livraison. Pour completer la fiche matériel, avec num serie, date de garantie ... il va falloir que les personnes qui l'ont commandé se reconnectent.
Une fois le materiel "validated", il ne peut plus etre supprimé. Il ne peut qu'etre sorti de l'inventaire.
Tous les personnels du labo peuvent demander une sortie d'inventaire, il n'y a que les gestionnaires (profil "administration") qui peuvent faire la sortie effective.
Les materiels restent en base avec le statut "Archived" mais ne sont plus visibles sur les listes de l'appli. Les gestionnaires peuvent voir la liste du materiel archivé.

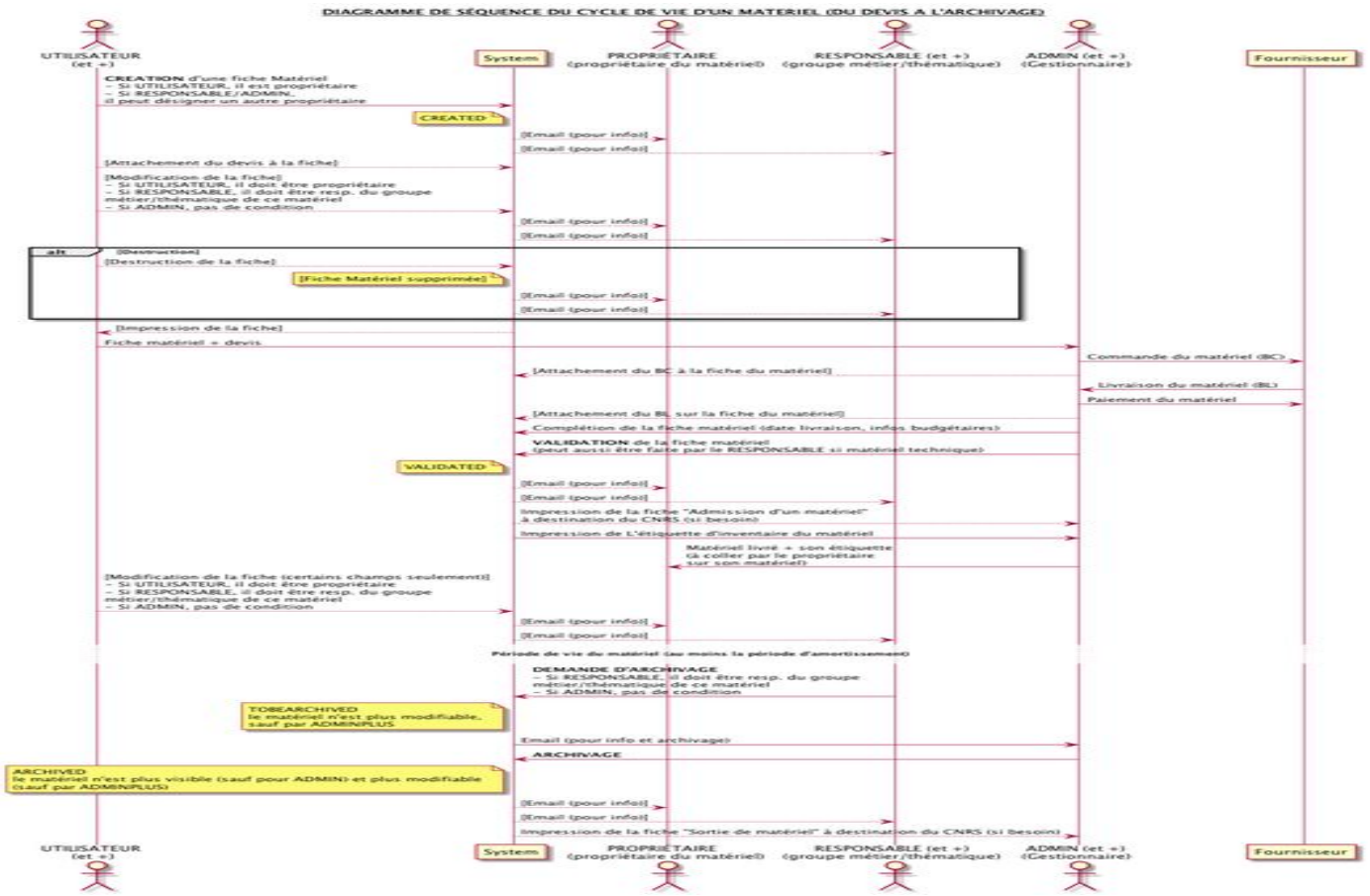
Il y a plusieurs profils dans l'appli :
User - utilisateur : Mr tout le monde. Il voit tous les materiels mais ne peut modifier que les siens.
Responsable : nous avions pense que les responsables de groupe pouvaient etre valideurs et avoir acces en modification a tous les materiels de leur groupe, mais chez nous, nous ne nous en servons pas. Il faudrait voir avec les autres labos s'ils se servent de ce profil et s'il est fonctionnel.
Administration : profil pour les gestionnaires. La gestion voit tous les materiels et peuvent tout modifier. Ils ont acces en plus a la partie administrative (bon de commande, EOTP, ...)
Super-administrateur: peut tout faire sauf sortir les materiels de l'inventaire.

2.2. Cycle de vie d'un matériel

2.2.1. Diagramme états-transitions



2.2.2. Diagramme de séquences



GESTION DES PROFILS UTILISATEURS (ACL)

Cette partie fait l’objet d’un document séparé

3. HOWTO

Ce document est un HOWTO, c'est à dire un guide technique pour savoir comment faire quoi.
Il donne des solutions à différents types de besoins ou problèmes, et explique aussi où trouver quoi.

3.1. LDAP

(updated 13/12/18)

Le ldap n'est appelé qu'au moment de la connexion d'un utilisateur :

1) LOGIN
src/Controller/UsersController.php : fonction login() :

```
$user = $this->LdapAuth->connection();
```

2) cela appelle :
src/Controller/Component/LdapAuthComponent.php : fonction connection() :

```
$resp = TableRegistry::get('LdapConnections')->ldapAuthentication($login, $password);
```

3) cela appelle :
src/Model/Table/LdapConnectionsTable.php : fonction ldapAuthentication()

```
if ($this->USE_LDAP) {
    if (strlen(trim($password)) == 0) return FALSE;
```

```
        $ldapConnection = ldap_connect($this->host, $this->port);
        ldap_set_option($ldapConnection, LDAP_OPT_PROTOCOL_VERSION, 3);
        if (@ldap_bind($ldapConnection, $this->authenticationType . '=' . $login . ',' . $this->baseDn, $password)) {
            return $this->getUserAttributes($login)[0];
        }
    }
```

3.2. Autres HOWTO plus anciens

Le contenu de ce chapître est à utiliser avec précaution car, étant assez ancien, il risque de ne pas être complètement utilisable (bien qu'il puisse encore l'être dans la plupart des cas). Cependant il reste utile pour information.

VERSION DE CAKEPHP
****
Version de cakephp ?
Configure::version(); // 2.1.3
Et aussi : fichier cakephp/lib/Cake/VERSION.txt

INSTALLATION
****
La procedure d'installation est decrite dans le fichier INSTALLATION.txt qui se trouve dans le dossier install/
Pour une installation à partir d'Eclipse, voir le document install/manual_install/INSTALLATION_MANUELLE_mode_expert.txt

CYCLE DE DEVELOPPEMENT A SUIVRE (11 commandements)
****

Je veux apporter un changement (correction ou evolution) au projet, comment dois-je faire ?

1) Incrementer la version du projet, a la fin du fichier cakephp/app/View/Layouts/default.ctp, et mettre à jour la date

2) Selectionner le changement a apporter (une demande) dans le Redmine du projet (https://projects.irap.omp.eu/projects/inventirap) :
- soit depuis la liste des demandes : onglet Demandes
- soit depuis la roadmap : onglet Roadmap, cocher la case "Anomalie", cliquer sur Appliquer, puis "version 1.3" (la version en cours depuis fin 2012)
Commencer de préférence par les "anomalies" parmi celles qui ont la plus haute priorité (et faire les "évolutions" dans un 2ème temps).
Choisir une demande et cliquer dessus pour aller sur sa fiche detaillee et voir le travail a faire.
Cliquer sur "mettre a jour", selectionner le statut "En cours", modifier eventuellement d'autres champs..., puis cliquer sur "Soumettre".
Noter l'URL de cette fiche (par exemple : https://projects.irap.omp.eu/issues/1050)
NB : Si la demande n'existe pas encore, la creer :
- cliquer sur l'onglet "Nouvelle demande"
- Selectionner "Anomalie" ou "Evolution"
- Positionner le statut a "En cours" si on veut travailler aussitot dessus (sinon, laisser a "Nouveau")
- Completer le reste de la fiche de demande (mettre "Assigne a" a "<<moi>>", choisir la version cible "version 1.3" ou "version 1.4", ...)
- cliquer sur le bouton "Creer"

3) Verifier que cette nouvelle demande apparait bien dans la roadmap (cliquer sur onglet "Roadmap", ..., puis version 1.3)

ÉTAPE OPTIONNELLE MAIS FORTEMENT CONSEILLÉE :
4) Ecrire un test qui vérifie que cette fonctionnalité ne marche pas encore (bug) ou bien n'est pas encore implémentée

On utilise ici l'approche TDD (Test Driven Development)
Ce test ne devrait pas passer (il est au rouge) ; il passera plus tard, quand on aura écrit le code nécessaire.
Bien sur, c'est dans la mesure du possible, car on ne peut pas TOUT tester.
Dans tous les cas, il faut ecrire un test qui s'approche le plus possible de la réalité à tester.
Voir pour cela la section "TESTS" (à la fin de ce document)

5) Faire les changements necessaires dans le code Php
Si ce changement implique aussi un changement dans la base de donnees,
copier le script SQL correspondant à ce changement dans un fichier database/update/db-update-YYYY-MM-DD.sql
portant la date du jour (voir des exemples dans database/update/old/).
Plus tard, il faudra penser à intégrer ce changement dans le script general de creation de la BDD database/BDD_IRAP.sql
(et/ou éventuellement les fichiers Insert_TablesFunct.sql, Insert_Users.sql, et Upd_TableConstraints.sql)
et donc deplacer le script de modification database/update/db-update-YYYY-MM-DD.sql dans database/update/old/db-update-YYYY-MM-DD.sql

6) Tester manuellement ce changement jusqu'a ce qu'il soit totalement OK
a) faire quelques tests manuels
b) Le test écrit à l'étape (4) doit maintenant passer (il est au vert).
Il doit être inclus dans l'ensemble des tests accessibles par le lien "AllTests".
c) Test de non régression : afin de s'assurer que cette modification du code n'entraîne aucune régression sur le reste du code,
tous les autres tests écrits avant doivent aussi passer (vert) : pour cela, exécuter l'url /test.php?case=AllTests

7) Compléter le fichier /README.txt, section HISTORIQUE DES VERSIONS (fichier situé à la racine du projet)
Y mettre le même commentaire que ce que tu mettras lors du commit

8) Mettre a jour TON code (en mode console, "svn update" depuis la racine du projet, ou bien depuis Eclipse, clic droit sur le projet, "Team/Update")
C'est important, au cas ou quelqu'un d'autre aurait fait des modifs (avant ou en meme temps que toi), et pour etre bien sur d'avoir la derniere version

9) Faire un "commit" du code, en collant dans le commentaire l'URL de la demande realisee (exemple : https://projects.irap.omp.eu/issues/1050)

10) Fermer la demande sur redmine
Sur la fiche detaillee, cliquer sur "Mettre a jour", changer le statut a "ferme", changer "% realise" a 100%, (copier l'URL de la fiche), cliquer sur Soumettre

11) Si c'est un changement important, le répercuter sur le site officiel
Le changement est important si c'est une anomalie ou bien si c'est une évolution attendue.
Demander alors au service informatique de le répercuter sur l'installation officielle (faire un "svn update", mail à loic.jahan@irap.omp.eu).
Si ce changement implique aussi la base de données, donner la directive que le fichier database/update/db-update-YYYY-MM-DD.sql doit être exécuté sur leur BDD.
Si ce changement impliquer une modification du fichier de configuration labinvent.php, donner la procédure à suivre dans le mail.

CORRIGER UNE ERREUR
RESOUDER UN PROBLEME
COMMENT DEBOGUER (DEBUG)
****
Cakephp vous affiche une erreur, mais vous ne savez pas d'ou vient le probleme.
Don't panic.
En general, un bon moyen de le savoir est de lire le fichier de log des erreurs
dans cakephp/app/tmp/logs/error.log

Dans le dossier logs, vous trouverez d'autres logs utiles :
- labinvent.log

- debug.log (pas sur que ce fichier soit encore utilisé...)

Remarque :
$this->log('Something broke');
==> écrit dans cakephp/app/tmp/logs/ :
- error.log
- inventirap.log

OU EST QUOI (WHERE IS WHAT) ?
****

- OU mettre a jour la VERSION du soft (AVANT CHAQUE COMMIT) ?
En attendant mieux, on fait cela a la fin du fichier cakephp/app/View/Layouts/default.ctp

- OU EST LA PAGE GENERALE (qui contient tous les elements) ? : cakephp/app/View/Layouts/default.ctp

- OU EST LA PAGE D'ACCUEIL ? : app/View/Pages/home.ctp

- OU SONT LES MENUS ? : app/View/Elements/
  - menu general + Recherche generale : menu.ctp
  - sous le menu general, et sous le champ "Recherche", titre du modele a ajouter : menu_index.ctp

- OU EST LA FEUILLE DE STYLE CSS ? : cakephp/app/webroot/css/inventirap.css

- OU EST DEFINIE LA PAGINATION ?

Dans le Controleur
Ex : la pagination des materiels est definie dans app/Controller/MaterielsController.php

```
    public $paginate = array(
            'limit' => 50,
            'order' => array('Materiel.id' => 'desc'));
```

- OU SONT LES INFOS CONCERNANT UNE ENTITE quelconque (Materiel, Emprunt, Suivi, Utilisateur) ?
par exemple, ou trouver les infos sur l'entite "Materiel" ? : Aller dans cakephp/app/
- le Modele : Model/Materiel.php
- le Controleur : Controller/MaterielsController.php
- les Vues (templates) : View/Materiels
    - Vue de consultation : scaffold.view.ctp
    - Vue d'ajout (add) et edition (edit) : scaffold.form.ctp
    - vue de liste : index.ctp

- OU SONT DEFINIS LES ROLES (ACL) ?
Dans app/Model/Utilisateur.php :

```
    private $acceptedRoles = array ('Utilisateur', 'Responsable', 'Administration', 'Super Administrateur');
    public function getAuthenticationLevelFromRole($role) {
            if ($role == 'Utilisateur')
                    return 1;
            elseif ($role == 'Responsable')
                    return 2;
            elseif ($role == 'Administration')
                    return 3;
            elseif ($role == 'Super Administrateur')
                    return 4;
```

```
        return 0;
    }
```

WORKFLOW des materiels
****
Le statut d'un materiel change selon le workflow suivant :
1) Un utilisateur lambda le cree (n'importe qui du labo) --> CREATED
2) L'Administration le valide (apres avoir eventuellement complete la fiche) --> VALIDATED
3) Un utilisateur lambda demande a l'archiver --> TOBEARCHIVED
4) L'Administration le sort de l'inventaire --> ARCHIVED

Notes :
- Dans l'ideal, le materiel est premierement cree par l'utilisateur concerne,
puis mis a jour par l'administration au moment de la commande (puis validé)
- L'administration peut toujours retrograder le statut d'un materiel (ce qui revient a annuler un changement de statut)

STRUCTURE GENERALE D'UNE PAGE WEB (TEMPLATE) = default.ctp
****
app/View/Layouts/default.ctp contient la structure suivante :

```
    <div id="container">
        <div id="header">
            LE HEADER AVEC SON LOGO
```

```
                <div class="user">
                        BIENVENUE $userName (ou invité)
                </div>
        </div>
        <div id="content">
                <?php echo $this->Session->flash(); ?>
                <?php echo $this->fetch('content'); ?>
        </div>
        <div id="footer">
                LE FOOTER
        </div>
</div>
```

La DIV "content" insère 2 contenus :
- le message flash éventuel (qui dit si une opération demandée s'est bien passée ou pas...)
- la section "content", qui n'est autre que le coeur de la page

La page d'ACCUEIL n'est qu'un "content" parmi d'autres.
Elle se trouve dans app/View/Pages/home.ctp
(elle est affichée par le controleur de pages nommé PagesController)

COMMENT VOIR LES REQUETES SQL FAITES PAR CAKEPHP ?
****

Dans le layout general (cakephp/app/View/Layouts/default.php), ajouter cette ligne :

```
<?php echo $this->element('SQL_DUMP'); ?>
```

Vous devez aussi vous assurer que vous etes bien en mode DEBUG=2 (et non pas 0 ou 1)
dans le fichier de configuration Config/labinvent.php (ou bien dans le fichier Config/core.php)

OU EST FAIT LE CHARGEMENT DU FICHIER CONFIG database.php ?
****

lib/Cake/Model/ConnectionManager.php (fonction _init())
Ce fichier fait partie de la librairie standard de CakePhp. Il ne faut donc pas y toucher.
A l'origine, j'avais (EP) changé ici le nom du fichier config "database.php" en "config.php",
ce qui marchait très très bien...
MAIS il ne vaut mieux pas modifier le comportement par défaut de cakephp,
car si un jour on change de version de cakephp, ça ne marchera plus.

CREATION ET GESTION DU NOUVEAU FICHIER DE CONFIG labinvent.php
****

(cf http://book.cakephp.org/2.0/fr/development/configuration.html#loading-configuration-files)

1) J'ai créé le nouveau fichier de config dans app/Config/ (par exemple labinvent.php)
Ce fichier doit au minimum contenir un tableau nommé $config :

```
$config = array(
        'labName' => 'IRAP',
        ...
);
```

2) Charger ce nouveau fichier de config au démarrage
Pour cela, ajouter cette ligne dans app/Config/bootstrap.php :

```
Configure::load('labinvent');
```

3) Lire (ou même modifier) les paramètres de ce fichier de config, depuis N'IMPORTE OU (controleur, modèle, vue) :

```
debug(Configure::version()); // pour afficher la version de Cakephp
debug(Configure::read()); // tout lire
debug(Configure::read('localisation')); // le tableau $localisation
$labName = strtoupper(Configure::read('localisation.labNameShort'));
debug(Configure::read('localisation.labName'));
if (Configure::read('USE_LDAP')) ...
if (Configure::read('debug') > 0 ) ...
```

On peut même modifier la valeur d'un paramètre dynamiquement comme ceci :

```
Configure::write('debug',2)
```

C'est d'ailleurs ce qui est fait dans app/Config/core.php

4) Penser à modifier le script d'installation install/installation.sh pour qu'il prenne en compte ce nouveau fichier

LOCALISATION (adapatation du logiciel à l'entité locale)
****

Vues (Pages) impactées :
- app/View/Layouts/default.ctp (template)

- app/View/Pages/home.ctp (Accueil)
- app/View/Pages/about.ctp (A propos)

Controleurs impactés :
- MaterielsController.php (fonction getLabNumber())

Documents impactés :
- Etiquette
- app/View/Documents/admission.ctp (Document d'admission)
- app/View/Documents/admission_cnrs.ctp (Document d'admission CNRS)
- app/View/Documents/admission_ups.ctp (Document d'admission UPS)

AJOUTER UNE NOUVELLE PAGE WEB
****
Par exemple, voici ce que j'ai fait pour ajouter la page "about", qui est accessible via l'url https://inventirap.irap.omp.eu/pages/about
1) Aller dans app/View/Pages/
1) Faire un copier/coller d'une page existante, par exemple infos.ctp, et l'appeler about.ctp
2) Remplir cette page avec le contenu souhaité
3) Tester que cette page est bien accessible via l'url http://.../pages/about
4) Ajouter un lien vers cette page, soit dans le menu outils (app/View/Pages/tools.ctp), soit dans le menu général (app/View/Elements/menu.ctp)
5) Si cette page ne doit pas être accessible a tout le monde, définir le profil minimum exigé dans app/Controller/PagesController.php, en ajoutant une ligne dans le tableau $minProfileAllowedForPage

LOGIN workflow
****

Controller/UtilisateursController/login() :
; cherche l'utilisateur dans la table utilisateurs pour voir si c'est un user SPECIAL ou bien un user quelconque
; appelle Controller/Component/LdapAuthComponent/connection() et getUserName($ldapAuthentication)

Controller/Component/LdapAuthComponent : contient les 2 méthodes
- connection() : appelle Model/LdapConnection/ldapAuthentication($login, $password)
- getUserName() : appelle Model/LdapConnection/getUserAttributes($ldapAuthentication)

Page d'accueil = View/Pages/home.ctp
(Controleur des pages simples = Controller/PagesController.php)

LOG
****

$this->log('Something broke');
==> écrit dans cakephp/app/tmp/logs/ :
- error.log
- inventirap.log

GOOGLE ANALYTICS INTEGRATION
****
adapté de http://blog.janjonas.net/2010-01-31/cakephp-google-analytics-integration

1) Copier ce code dans app/View/Elements/google-analytics.ctp :

```
<?php
$gaCode = Configure::read('google-analytics.tracker-code');
if ($gaCode) {
$googleAnalytics = <<<EOD
<script type="text/javascript">
  (function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
  (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o),
  m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m)
  })(window,document,'script','//www.google-analytics.com/analytics.js','ga');

  ga('create', 'UA-45668893-3', 'omp.eu');
  ga('send', 'pageview');

</script>
EOD;
echo $googleAnalytics;
}
?>

/* ANCIEN CODE JAVASCRIPT :
<script type="text/javascript">
var gaJsHost = (("https:" == document.location.protocol) ? "https://ssl." : "http://www.");
```

```
document.write(unescape("%3Cscript src='" + gaJsHost + "google-analytics.com/ga.js' type='text/javascript'%3E%3C/script%3E"));
</script>
<script type="text/javascript">
try {
var pageTracker = _gat._getTracker("$gaCode");
pageTracker._trackPageview();
} catch(err) {}</script>
*/
```

2) Inclure le view element dans app/View/Layouts/default.ctp (juste avant </body>):

```
<?php echo $this->element('google-analytics'); ?>
```

3) Définir le tracker code dans la configuration (/app/Config/core.php):

```
Configure::write('google-analytics.tracker-code', false); // disables Google Analytics
Configure::write('google-analytics.tracker-code', 'YOUR-TRACKING-CODE'); // enables Google Analytics
```

GENERALITES

****

- URL : http://localhost/inventirapsvn/cakephp/

- BD admin : http://localhost/phpmyadmin

- LOG : cakephp/app/tmp/logs/inventirap.log

- (UPDATE : cd Inventirap/ ; chmod -R 777 ./cakephp/app/tmp/)

- (SORTIE INVENTAIRE a faire : pc portable DELL (ex Nicolas Andre) - sept 2012 CNES 4017 (carte mere hs))

- Eclipse config :
Setup syntax highlighting for .ctp files:
Window > Preferences > General > Appearance > content Types > Text > PHP Content type > Add.. , then put in *.ctp.

- Cakephp config : cakephp/app/Config/
Mode debug ON (pour voir les erreurs et pour vider le cache en cas de changement sur la BD)
core.php : Configure::write('debug', 1);

- Fichier de demarrage :
cakephp/index.php (qui pointe sur cakephp/app/webroot/index.php)
(Attention, il y a aussi un fichier cakephp/app/index.php qui pointe sur webroot/index.php)
(ROOT doit pointer sur le dossier cakephp/)
Le fichier execute ensuite est cakephp/lib/Cake/bootstrap.php qui lance ./Core/App.php
and so on...

RECHERCHE de materiels
****
3 methodes :
- recherche generale (sous le menu general) : cakephp/app/View/Elements/menu.ctp

- VUE de recherche : cakephp/app/View/Materiels/find.ctp
- ACTION de recherche : cakephp/app/Controller/MaterielsController (methode find())

ACL (Controle d'acces aux ressources)
****
NB: Cette section n'est sans doute plus très à jour ; sur ce sujet, il est préférable de lire le document docs/userguide/ACL.doc (ou .pdf)

Synthese sur les droits selon le profil

Profils (roles), dans le sens du pouvoir croissant :
Qq = Utilisateur Quelconque (lambda)
Rp = Responsable
Ad = Administration
Sa = Superadmin

Actions :
C = Create
R = Read (voir, consulter)
U = Update (mettre a jour)
D = Delete (supprimer)
V = Valider un materiel CREATED --> passe alors en statut VALIDATED
A = Demander l'Archivage d'un materiel VALIDATED --> passe alors en statut TOBEARCHIVED
S = Sortir de l'inventaire (Valider une demande d'archivage d'un materiel TOBEARCHIVED) --> passe alors en statut ARCHIVED
E = Exporter

Par defaut, le superadmin a acces a TOUT

Materiels :
- Qq a les droits C, R (sauf champs admin), U (si createur et sauf champs admin), A, D (si CREATED et owner)
- Rp a les droits C, R (sauf champs admin), U (sauf champs admin), D (si CREATED), V, A, E
- Ad a les droits C, R, U (ssi NOT ARCHIVED), D (si CREATED), V, (A mais inutile car fait directement S sans passer par A), S, E

Suivis et Emprunts :
- Dans tous les cas, on ne doit pas pouvoir emprunter ou suivre un materiel non valide (CREATED)
- Qq a les droits C, R, U (si createur), D (si createur)
- Rp a les droits C, R, U, D
- Ad a les memes droits que Rp

VUES specifiques :
- Acces aux Outils : reserve a Rp et Ad (vue contenant des liens vers differentes ressources telles que utilisateurs, materiels, categories...)
- L'administration a une vue resumee sur la page d'accueil (liens directs vers actions a faire)
- L'administration a une vue enrichie de la liste des materiels :
  - filtres (y-compris sur ARCHIVED)
  - export en CSV (pour tableur Excel)
  - upgrade du statut (validation et sortie)

Autres regles (de gestion) importantes :
- un materiel non valide (CREATED) peut ���tre supprime uniquement par le createur de la fiche, le responsable (Roger), et l'administration
Idem pour la mise ��� jour de cette fiche
- un materiel valide (VALIDATED) n'est pas supprimable
- les champs ADMINISTRATIFS d'un materiel ne sont visibles et modifiables que par l'administration
- par defaut, la liste des materiels affiche tous les materiels SAUF ceux qui sont sortis de l'inventaire (ARCHIVED) qui sont donc masques.

Il est de toutes facons toujours possible (pour l'administration seulement) de les voir, grace au nouveau filtre "Archives" present sur la liste des materiels
- la recherche doit s'effectuer dans TOUS les materiels (y-compris ARCHIVED)

COMMENT CONTOURNER LE LDAP officiel
****
(ATTENTION: CONTENU OBSOLETE A METTRE A JOUR (EP))

(cf Controller/UtilisateursController.php : methode login())

1) Si on veut tester le LDAP, on peut utiliser le LDAP de la Virtual Machine Upsilon (CentOS 6)
Dans ce cas, il faut ajouter dans la BD les utilisateurs specifiques suivants :
Ajout d'utilisateurs de TEST (LDAP upsilon)
INSERT INTO utilisateurs (nom, login, email, role, groupes_metier_id) VALUES
('Hillembrand Cedric', 'Cedric', 'Cedric.Hillembrand@irap.omp.eu', 'Super Administrateur', 1),
('Turner Daniel', 'Daniel', 'Daniel.Turner@irap.omp.eu', 'Administration', 1),
('Sky Gin', 'Gin', 'Gin.Sky@irap.omp.eu', 'Responsable', 1),
('Robert Henri', 'Henri', 'Henri.Robert', 'Utilisateur', 1);

2) Sinon, le plus simple est d'utiliser un FAUX (fake) LDAP interne a l'application
Dans ce cas, il faut decommenter la variable $fakeLDAP dans le fichier config.php et ajouter dans la BD les utilisateurs specifiques suivants :
Ajout d'utilisateurs de TEST (hors LDAP)
INSERT INTO utilisateurs (nom, login, email, role, groupes_metier_id) VALUES
('Hubert SuperAdmin', 'superadmin', 'hubert.superadmin@irap.omp.eu', 'Super Administrateur', 1),

```
        ('Pierre Responsable', 'responsable', 'pierre.resp@irap.omp.eu', 'Responsable', 2),
        ('Jean Administration', 'admin', 'Jean.Administration@irap.omp.eu', 'Administration', 5),
        ('Jacques Utilisateur', 'user', 'Jacques.Utilisateur@irap.omp.eu', 'Utilisateur', 7);
```

Informations sur le LDAP :

Classe LdapAuthComponent (extends AuthComponent), definie dans : app/Controller/Component/LdapAuthComponent.php

Definition des roles (ACL) : app/Model/Utilisateur.php :

```
    private $acceptedRoles = array ('Utilisateur', 'Responsable', 'Administration', 'Super Administrateur');
    public function getAuthenticationLevelFromRole($role) {
        if ($role == 'Utilisateur')
            return 1;
        elseif ($role == 'Responsable')
            return 2;
        elseif ($role == 'Administration')
            return 3;
        elseif ($role == 'Super Administrateur')
            return 4;
        return 0;
    }
```

Lecture du fichier de config : app/Model/LdapConnection.php :

```
        private function checkConfiguration()
```

Workflow du LDAP :

1) Page d'accueil : j'entre mon login et pwd

2) clic sur bouton "Se Connecter" --> execute l'action Utilisateurs/login (dans app/Controller/UtilisateursController.php)

COMMENT AJOUTER UNE NOUVELLE TABLE DANS LA BD
****

On explique ici ce qu'il faut faire quand on veut ajouter une nouvelle table dans la base de données,
table qui doit être prise en compte dans l'application.
Cette explication est donnée avec 2 exemples.

1) Premier exemple : ajout de la table sur-categorie (EP)

Avant cet ajout, il n'y avait que la table categorie et la table sous-categorie.
Vous trouverez plus de détails sur ce sujet dans la section suivante nommée "COMMENT J'AI FAIT POUR AJOUTER UN 3eme niveau de categorie appele sur_categorie (ou domaine)"

a) impact sur la BD
- ajout d'une nouvelle table sur_categories(id,nom) :

```
CREATE TABLE IF NOT EXISTS sur_categories (
  id int(11) NOT NULL AUTO_INCREMENT,
  nom varchar(45) DEFAULT NULL,
  PRIMARY KEY (id),
  UNIQUE KEY nom_UNIQUE (nom)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1;
```

- ajout dans la table categories d'une cle etrangere sur_categorie_id :

```
ALTER TABLE categories
    ADD sur_categorie_id INT( 11 ) NULL DEFAULT NULL,
      ADD CONSTRAINT fk_sur_categorie_id FOREIGN KEY (sur_categorie_id) REFERENCES sur_categories (id) ON DELETE NO ACTION ON
UPDATE NO ACTION;
```

- ajout dans la table materiels d'une cle etrangere sur_categorie_id :

```
ALTER TABLE materiels
    ADD sur_categorie_id INT( 11 ) NOT NULL after designation,
    ADD CONSTRAINT fk_materiels_sur_categorie_id FOREIGN KEY (sur_categorie_id) REFERENCES sur_categories (id) ON DELETE NO
ACTION ON UPDATE NO ACTION;
```

(attention, il faut d'abord vider les lignes des tables suivis, emprunts, et materiels)

```
delete from suivis;
delete from emprunts;
delete from materiels;
```

Il faut aussi ajouter des sur-categories :

```
# Ajout de quelques sur-categories
insert into sur_categories (id,nom) values
(1,'Electronique'),
(2,'Informatique'),
(3,'Instrumentation')
(4, 'Logistique'),
(5, 'Mecanique'),
(6, 'Optique')
```

;

Relier toutes les categories a une sur-categorie (la 1)
update categories set sur_categorie_id=1 where id<10;
update categories set sur_categorie_id=2 where id>=10 and id<20;
update categories set sur_categorie_id=3 where id>=20;

b) impact sur les modeles (cakephp/app/Model)
- Ajout du nouveau modèle SurCategorie.php : copie de Categorie.php avec "hasMany categorie"
- Modif du modèle Categorie.php : + belongsTo="SurCategorie"

c) impact sur les controleurs (cakephp/app/Controller)
- Ajout du nouveau controleur SurCategoriesController.php : minimaliste (comme CategoriesController)
- Modif du controleur CategoriesController.php : +getBySurCategorie()

d) impact sur les vues (cakephp/app/View)
- SurCategorie : no view (scaffold ?)
- Categorie : actuellement no view (scaffold ?)
--> creer une vue (comme SousCategorie)
+ get_by_surcategorie.ctp
+ scaffold.form.ctp

e) View/Pages/tools.ctp : ajouter une entree au menu pour les Sur-Categories

f) impact sur TOUTES les vues de Materiel
add/edit/view/index

dans la vue index, remplacer la categorie par la sur-categorie
GROS BOULOT sur la vue ADD/EDIT (+ javascript)

2) Deuxième exemple : ajout de la table type_suivis (VM)

Un peu plus haut est expliqué comment créer une table, je vais ici expliquer comment créer et remplir tout ce qui est basiquement nécessaire pour l'utiliser, en prenant pour exemple la table type_suivis.

Après avoir créé la table, on lui crée :
- Un controller : Héritant de AppController, avec une variable $scaffold sui se chargera de presque tout et une variable $name avec son nom.
- Un model : Avec la même variable $name et une variable $displayField qui correspond a la designation dans la BDD (ex: "nom");
  Il doit contenir la fonction typeSuiviNameIsUnik($name) {} (qu'on retrouve dans sites ou organisme) ainsi que le $validate comprenant les validations necessaires.
- Une vue, selon l'utilité, n'est pas forcément nécessaire grâce au scaffold.

Par la suite, pour lister son contenu via un lien dans "outils" par exemple, il suffit de créer le chemin dans view/pages/tools.ctp.

CREER UN CHAMP DATE (VM)
****

Pour expliquer comment créer un champ date fonctionnel, je vais prendre l'exemple de la Date de reception.
Une fois vôtre colonne créé dans la table marteriel, vous devrez faire des modifications dans :

MaterielController : //Passer la date au format français

```
if(isset($this->request->data['Materiel']['date_reception'])){
$this->request->data['Materiel']['date_reception'] = date("d-m-Y", strtotime($this->request->data['Materiel']['date_reception'])); }
```

le Model Materiel : Créer un champ de validation adapté à la date, avec un regex. Et surtout remplir le formatage de date à l'américaine :

```
if (isset($this->data['Materiel']['date_reception'])) {
        $originalDate = $this->data['Materiel']['date_reception'];
        $this->data['Materiel']['date_reception'] = date("Y-m-d", strtotime($originalDate));
}
```

La vue Materiel : - Scaffold.view : Repasser la date en français et l'afficher.
- Scaffold.form : Créer le champ date du formulaire

Puis il faut adapter le champ date reception presque partout ou il y a le champ date acquisition.

COMMENT J'AI FAIT POUR AJOUTER UN 3eme niveau de categorie appele sur_categorie (ou domaine) (EP) ?
****

je me rends compte d'une incoherence de stockage dans la table Materiels.
En effet, quand on saisit un materiel, on doit choisir une categorie ET une sous-categorie.
Du coup, les 2 id (categorie + sous-categorie) sont stockes dans la table Materiels...
Or, c'est inutile car la sous-categorie suffit a determiner la categorie...
Cette redondance pourrait meme amener des incoherences dans la table Materiels si par exemple on fait des modifications

dans les tables categories et sous_categories sans les repercuter dans la table materiels !!
Je suppose que c'est un choix de facilite qui a ete fait par upsilon.

Donc, si vous etes ok, et a moins que Upsilon me dise qu'il y avait une bonne raison de faire ce choix (j'en doute),
je propose de modifier le code pour que seule la sous-categorie soit stockee (on ne stocke plus la categorie).

En plus, cela simplifiera l'ajout du 3eme niveau "sur-categorie" puisque lui-meme n'aura pas besoin d'etre stocke etant donne qu'il est automatiquement determine par la categorie.
On aura alors :
sous_categorie_id -> categorie_id -> sur_categorie_id
Avec seulement la sous_categorie, on pourra determiner la categorie, et donc la sur_categorie.

Du coup, si je fais cette modif, on pourrait meme se permettre de demarrer officiellement INVENTIRAP rapidement.
En effet, l'ajout de la sur-categorie ne change rien au contenu des tables "administratives" (materiels, emprunts, suivis),
et donc on pourra le faire plus tard, meme apres que l'administration ait commence a remplir la BD.
Ca sera totalement transparent.

1) suppression de la redondance (categorie_id) dans la table materiels

a) impact sur la vue index de Materiel
add/edit/view/index

2) ajout d'une sur-categorie

a) impact sur la BD
- ajout d'une nouvelle table sur_categories(id,nom) :
CREATE TABLE IF NOT EXISTS sur_categories (

```
  id int(11) NOT NULL AUTO_INCREMENT,
  nom varchar(45) DEFAULT NULL,
  PRIMARY KEY (id),
  UNIQUE KEY nom_UNIQUE (nom)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1;
```

- ajout dans la table categories d'une cle etrangere sur_categorie_id :

```
ALTER TABLE categories
    ADD sur_categorie_id INT( 11 ) NULL DEFAULT NULL,
      ADD CONSTRAINT fk_sur_categorie_id FOREIGN KEY (sur_categorie_id) REFERENCES sur_categories (id) ON DELETE NO ACTION ON
UPDATE NO ACTION;
```

- ajout dans la table materiels d'une cle etrangere sur_categorie_id :

```
ALTER TABLE materiels
    ADD sur_categorie_id INT( 11 ) NOT NULL after designation,
    ADD CONSTRAINT fk_materiels_sur_categorie_id FOREIGN KEY (sur_categorie_id) REFERENCES sur_categories (id) ON DELETE NO
ACTION ON UPDATE NO ACTION;
```

(attention, il faut d'abord vider les lignes des tables suivis, emprunts, et materiels)

```
delete from suivis;
delete from emprunts;
delete from materiels;
```

Il faut aussi ajouter des sur-categories :

```
# Ajout de quelques sur-categories
insert into sur_categories (id,nom) values
(1,'Electronique'),
```

(2,'Informatique'),
(3,'Instrumentation')
(4, 'Logistique'),
(5, 'Mecanique'),
(6, 'Optique')
;

Relier toutes les categories a une sur-categorie (la 1)
update categories set sur_categorie_id=1 where id<10;
update categories set sur_categorie_id=2 where id>=10 and id<20;
update categories set sur_categorie_id=3 where id>=20;

b) impact sur les modeles (cakephp/app/Model)
- SurCategorie.php : copie de Categorie.php avec "hasMany categorie"
- Categorie.php : + belongsTo="SurCategorie"

c) impact sur les controleurs (cakephp/app/Controller)
- SurCategoriesController.php : minimaliste (comme CategoriesController)
- CategoriesController.php : +getBySurCategorie()

d) impact sur les vues (cakephp/app/View)
- SurCategorie : no view (scaffold ?)
- Categorie : actuellement no view (scaffold ?)
--> creer une vue (comme SousCategorie)
+ get_by_surcategorie.ctp
+ scaffold.form.ctp

e) View/Pages/tools.ctp : ajouter une entree au menu pour les Sur-Categories

f) impact sur TOUTES les vues de Materiel
add/edit/view/index
dans la vue index, remplacer la categorie par la sur-categorie
GROS BOULOT sur la vue ADD/EDIT (+ javascript)

TESTS
****

A - EXECUTION
****

Prérequis : PHPUNIT 3 doit être déjà installé et accessible depuis la console (phpunit -version) via le fichier php.ini
(ATTENTION : cakephp 2.x n'est pas compatible avec PhpUnit 4)

Avec XAMPP (conseillé) : PhpUnit 3 est déjà inclus
Avec MAMP (+ difficile) : pour installer PhpUnit, suivre cette documentation :
http://www.dolinaj.net/software-installation/mac/how-to-install-phpunit-with-mamp-on-mac/

Procédure à suivre pour pouvoir exécuter les tests unitaires et fonctionnels livrés avec le logiciel :

1) Ajouter une nouvelle configuration de base de données dans votre fichier app/Config/database.php pour la BD de test

Exemple de configuration :

```
public $test = array(
        'datasource' => 'Database/Mysql',
        'persistent' => false,
        'host' => 'localhost',
        'database' => 'test_labinvent',
        'login' => 'root',
        'password' => '',
);
```

2) Créer la BD de test (vide)
A l'aide de phpmyadmin ou bien avec un client mysql quelconque,
créer une BD de test vide que vous nommerez avec le même nom que celui donné dans la configuration ci-dessus,
soit pour l'exemple, "test_labinvent"
Syntaxe SQL : "create database test_labinvent"

3) Passer en mode "debug"
Dans votre fichier de configuration labinvent.php, mettez votre paramètre "debug" à 1 ou 2 (mais pas à 0) :

```
        'debug' => 2,
```

4) Se connecter à l'application
Les test ne passeront pas si vous n'êtes pas connectés

5) Exécuter les tests

a) Exécution depuis l'application (conseillé) :
ATTENTION : vous devez être logué pour que les tests passent !!!
Il suffit d'aller à l'URL /test.php de votre installation du logiciel LabInvent
(vous pouvez aussi essayer l'URL "/app/webroot/test.php", ou encore "/cakephp/test.php")
Cette page de tests se trouve dans cakephp/app/webroot/
Vous avez alors accès à 2 types de tests :
- App : Tests ==> les tests écris pour tester l'application Labinvent
- Core : Tests ==> les tests fournis avec cakephp pour tester le framework
Pour exécuter tous les tests liés à l'application Labinvent (à faire systématiquement avant de commiter tout changement) :
cliquer sur "Tests" sous "App", puis sur "AllTests"
("AllController" exécute tous les tests de controleurs ; "AllModel" exécute tous les tests de modèles)

b) Execution depuis la console :
Aller dans le repertoire app/
Pour tester le controleur MaterielsController :
./Console/cake test app Controller/MaterielsController

B - ECRITURE DE NOUVEAUX TESTS
****

cf http://book.cakephp.org/2.0/en/development/testing.html

Aller dans app/Test/

Ce dossier contient l'arborescence suivante :

- Case/ : les tests
    - Controller/ : les tests de controleurs
    - Model/ : les tests de modèles
    - View/ (peu ou pas utilisé) : les tests de vues (ou de helpers)
    - AllControllerTest.php : exécution de tous les tests de controleurs
    - AllModelTest.php : exécution de tous les tests de modèles
    - AllTestsTest.php : éxécution de TOUS les tests

- Fixture/ : les différentes initialisations nécessaires dans la BD de test pour pouvoir executer les tests
Ces "fixtures" sont automatiquement executees AU DEBUT de chaque test.
Ce dossier contient un fichier pour chaque table pour laquelle on a besoin d'une "fixture".

1) Les "fixtures"

La façon la plus basique de creer une fixture pour une table donnee est de la realiser automatiquement a partir d'une copie de la table de la vraie BD.
Par exemple, pour que la table "categories" de la BD de test contienne la meme chose que la table de la vraie BD, il suffit de creer un fichier CategorieFixture.php contenant ceci :

```
class CategorieFixture extends CakeTestFixture {
    public $import = array('model' => 'Categorie', 'records' => true);
}
```

Au demarrage des tests, cette table sera chargee automatiquement avec les vraies donnees.
A la fin des tests, cette table sera vidée.

Dans le cas particulier de la table "materiels", on prefere l'initialiser nous-memes avec des valeurs choisies.

Exemple d'une fixture avec 2 materiels (dans le fichier MaterielFixture.php) :

```
class MaterielFixture extends CakeTestFixture {

    public $import = 'Materiel'; // import only structure, no record

    public $records = array(
                array(
                            'designation' => 'matos1',
                            'sur_categorie_id' => 1,
                            'categorie_id' => 11,
                            'materiel_administratif' => 0,
                            'materiel_technique' => 1,
                            'status' => 'CREATED',
                            'nom_createur' => 'Pallier Etienne',
                            'nom_modificateur' => 'Jean Administration',
                            'nom_responsable' => 'Jacques Utilisateur',
                            'email_responsable' => 'Jacques.Utilisateur@irap.omp.eu',
                ),
                array(
                            'designation' => 'matos2',
                            'sur_categorie_id' => 1,
                            'categorie_id' => 11,
                            'materiel_administratif' => 0,
                            'materiel_technique' => 1,
                            'status' => 'CREATED',
                            'nom_createur' => 'Pallier Etienne',
```

```
                                'nom_modificateur' => 'Jean Administration',
                                'nom_responsable' => 'Jacques Utilisateur',
                                'email_responsable' => 'Jacques.Utilisateur@irap.omp.eu',
                        ),
    );
}
```

2) Les tests

Prenons l'exemple des tests ecrits pour le controleur des materiels (MaterielsController). Il devra s'appeler MaterielsControllerTest et aura la structure suivante :

```
class MaterielsControllerTest extends ControllerTestCase {

    // Liste des fixtures à charger avant l'execution des tests
    public $fixtures = array('app.materiel', 'app.sur_categorie', 'app.categorie', 'app.sous_categorie',
                    'app.groupes_thematique', 'app.groupes_metier', 'app.suivi', 'app.emprunt'
    );

    // Initialisations diverses a faire avant chaque test
    public function setUp() {
            parent::setUp();
    }

    // un 1er test
    public function testMonPremier() {
```

```
        $result = $this->testAction(...);
        $this->assert...('resultat attendu', $result);
    }

    // un 2eme test
    public function testMonDeuxieme() {
        $result = $this->testAction(...);
        $this->assert...('resultat attendu', $result);
    }

    ...

}
```

Voir le vrai fichier Test/Case/Controller/MaterielsControllerTest.php

3) Execution

Exemple avec l'execution du test MaterielsControllerTest

a) Execution depuis le site web :
/test.php?case=Controller%2FMaterielsController
Ajouter &debug=1 à l'url pour voir tous les messages de debug

b) Execution depuis la console :
Dans le repertoire app : ./Console/cake test app Controller/MaterielsController

Ajouter --debug pour voir tous les messages de debug

DATE PICKERS
****

Pour fontionner, le datePicker fait appel dans la page "View/Layout/default" à 3 scripts (jquery-1.5.2.js, jquery-1.8.12.js, DatepickerConfig.js) présents dans le repertoire "webroot/js/" et à un fichier "Theme" (smoothness.css) présent dans le repertoire "webroot/css/"

Le thème global peut très facilement être changé en téléchargeant le fichier css de son choix, à cette adresse: "http://jqueryui.com/themeroller/" et en remplaçant celui se trouvant dans "webroot/css/"

Les options du datePicker sont modifiables dans le fichier "webroot/js/DatepickerConfig.js" et sont assez explicites.
Malgré cela, pour plus de précisions, la doc est facilement consultable à cette adresse: "http://jqueryui.com/datepicker/"

4. DOCUMENTATION UTILISATEURS

Cette doc est en cours de migration à l’intérieur du présent document.

En attendant, voici le lien vers la doc d’origine

5. DOCUMENTATION TECHNIQUE (POUR LES DEV)

Cette doc est en cours de migration à l'intérieur du présent document.

En attendant, voici les liens vers la doc d'origine :

- Doc technique
- Doc de développement

5.1. Modèle de données (Data model)

Schéma de la base de données (v2.6) :