# Développement NOALYSS

# Table des matières

# Noalyss 9.3, 2024-09-01

# Chapitre 1. Développement de NOALYSS

## 1.1. Intro

Au début, les programmeurs développent la totalité d'un produit , ils regroupent des fonctions dans des librairies. Plus tard, ils vont non seulement réutiliser les fonctions qu'ils ont déjà développées mais ils adoptent aussi une méthode de développement, pour le nommage des fonctions, les conventions sur le code,... afin d'avoir un code homogène.

Et ces librairies, scripts , conventions finissent par devenir un cadre de travail (framework) qui permet d'avoir un code homogène et plus facilement abordable pour les nouveaux venus sur le projet.

NOALYSS a déjà plus de 22 ans , il est né bien avant la naissance des framework pour PHP, il était déjà fort avancé quand est apparu symphony.

NOALYSS, en 2002, était développé avec vi ou emacs, sous CVS pour la gestion des versions, parce qu'il n'y avait pas d'outils professionnels pour travailler avec PHP , même pas un débuggueur, ou des suites de tests.

Aujourd'hui, les développements en PHP sont supposés utiliser les normes PSR0/4, mais cela imposerait une énorme réécriture de NOALYSS, ces normes sont arrivées bien après la création de NOALYSS.

D'un autre côté, le code est parfois réécrit et on essaie d'avoir un code homogène.

Cela explique que l'organisation des fichiers et la façon de travailler avec NOALYSS , ne ressemble pas aux framework existants.

## 1.2. Organisation des fichiers

- html ce qui est appelable directement (modèle mvc)
- include/ fichier inclu par html (*.inc.php)
- include/ajax traitement pour les réponse ajax
- include/template : template , écran & code html
- include/lib : les classes librairies
- include/class : class métiers
- include/database : mapping table, vues et object [ORM]
- include/export : fichier pour confection de documents qu'on envoie (zip , csv, pdf ...)
- include/ext : plugin(alias extensions)
- include/sql : répertoire avec les scripts SQL pour les mises à niveau du programme

## 1.3. Convention

### 1.3.1. Nommage

Les fichiers dans

**include**

sont destinés à être inclus et exécuter depuis le 1er script (route)

```
<action>.inc.php
```

**ajax**

Appelé depuis une procédure javascript , le nom de fichier est

```
ajax_<action>
```

### 1.3.2. database

mapping table, vues et object

```
<table>_SQL.class.php
```

### 1.3.3. template

template , écran & code html

```
<nom_class>-<nom-fonction>.php
```

### 1.3.4. Outil

Tout doit être documenté avec Doxygen

```
référence  : https://www.doxygen.nl/
```

Tous les objets SQL doivent aussi être documentés

```
référence : https://www.postgresql.org/docs/current/sql-comment.html
```

La documentation doit être au format asciidoc , ou markdown dans le répertoire doc

```
référence : https://asciidoctor.org/
```

Utilisation de phpunit ou autre système (fichiers de test avec assert() ) pour faire des tests unitaires

```
référence : https://phpunit.de/index.html
```

## 1.4. Débuggage

Afin de débugger, il suffit dans le fichier `noalyss/include/config.inc.php` d'ajouter

```
define ("DEBUGNOALYSS",2);
```

Le niveau de déboggage est :

**0**
: production, aucune erreur n'est affichée

**1**
: les erreurs et les avertissements sont affichées

**2**
: affichage comme le niveau 1 , mais en plus des informations sur la taille de l'écran (bande de couleur en haut), pour aider la mise au point des pages responsives , affichage des fichiers inclus, d'objets,...

Pour le niveau 2 , on utilisera la class `Noalyss\Dbg` voir https://wiki.noalyss.eu/doc/class_noalyss_1_1_dbg.html

*Exemple 1. Fonctions de débuggages:*

Affiche le nom du fichier inclus

```
\Noalyss\Dbg::echo_file(__FILE__);
```

Si DEBUGNOALYSS est supérieure à 1 , affiche le contenu de `row` dans un élément cacheable

```
if ( DEBUGNOALYSS>1) echo \Noalyss\Dbg::hidden_info("\$row", $row);
```

Si DEBUGNOALYSS est supérieure à 1, affiche l'utilisateur courant

```
 \Noalyss\Dbg::echo_var(1,sprintf("current user is [%s]",$cn->get_value("select
current_setting('noalyss.user_login')")));
```

| | |
|---|---|
| **NOTE** | l'utilisateur connecté, est sauvé dans la mémoire de la base de données et peut de ce fait être utilisé dans des triggers, voir postgresql : current_setting |

# 1.5. Principe : développement en MVC

Le développement en MVC est un développement qui suit les concepts Modèle Vue Contrôle. Il en existe 2 versions. Ici, je ne parlerai que de la première.

En pratique, je crée 3 répertoires de base /html et /include et /include/template, le premier contient les « contrôles » le second les «vues» et le troisième les « modèles ».

## 1.5.1. Contrôle

Un contrôle est une page php, qui en fonction de l'action demandé va inclure la page «modèle» qu'il faut.

Dans noalyss, cette page est :

**do.php** pour toutes les demandes de pages

**export.php** pour toutes les impressions (PDF, CSV, ...)

**ajax_misc.php** pour l'ajax.[1]

Exemple : la page contrôle contient

```
<?php

if ( !isset ($_REQUEST['do']))  exit();

$do=$_REQUEST['do'];

if ( $do == 'auteur' ) {

    require_once('include/auteur.inc.php');

}

?>
```

L'explication est assez simple, si la page n'est pas appelée avec une variable do, donnée par un POST ou GET alors, le traitement s'arrête. Sinon en fonction de la valeur de la variable, il va inclure le fichier **vue**. Par habitude, mes fichiers vues ont toujours l'extension .inc.php.

Dans noalyss, le fichier do.php va chercher la vue à intégrer dans la base de données, ce qui améliore la sécurité mais aussi une grande souplesse pour les menus. Facile d'en ajouter, d'en retirer, de les renommer...

NOTE : Puisque le fichier à inclure est dans la base de données, si votre profil ne l'inclus pas, alors

vous avez pas du tout accès à ce menu, cela s'étend aux impressions, il est possible d'empêcher les impressions. Les menus et profils se configurent dans [C1MENU]

### 1.5.2. Vue

Grâce à l'exemple vu dans Contrôle, vous avez compris ce qu'est le concept vue: c'est la présentation de votre action. Donc pour continuer l'exemple, dans le document include/auteur.inc.php, on aurait

```
<?php

require_once ('include/class_auteur.php');

if ( ! isset($_REQUEST['sdo']) ) exit();



$sdo=$_REQUEST['sdo'];



if ( $sdo == 'add' ) {

    $auteur=new Auteur();

    $auteur->display_form();

}

?>
```

Ici, comme dans contrôle, on vérifie qu'une action est bien demandée, si c'est la cas, on appelle un objet et on lui demande d'afficher quelque chose. Cet objet est ce qui est dans le concept MVC, un modèle. En fait, on dirait une espèce de sous-contrôle

### 1.5.3. Modèle

Eh oui, le dernier est donc le modèle. Ici ce sont deux classes, l'une va se charger de sauver, afficher, effacer les données et gère donc les relations avec la base de données et l'autre l'objet métier.

Le nom de ce fichier doit être

```
<nom de la classe>-<fonction>.php
```

Ce n'est pas toujours le cas, et cette réécriture se fait au fur et à mesure.

Maintenant que vous avez compris MVC, je vais développer un peu les modèles.

Cette partie est un peu plus complexe. Tout d'abord, personnellement, je la divise en deux parties, la première doit gérer les données de la base de données et l'autre les traiter. Comment y arriver ?

Il est vraiment fastidieux de construire une chaîne de caractère contenant tout le code html, surtout quand on emploie aussi du javascript et des tableaux. La solution que j'ai trouvé, je l'ai trouvé en m'inspirant de l'idée de smarty. En fait, je crée un sous-répertoire dans include que j'appelle template, et dedans je mets le code HTML de ce qui doit être affiché

```
function display_form() {

ob_start();

include  'template/auteur_display_form.php';

$r=ob_get_contents();

$ob_clean();

return $r;



}
```

exemple Ici on inclus le fichier, comme il est en php, il sera interprété par PHP, il contient surtout du code HTML et des balises PHP, on peut aussi manipuler la chaîne retournée avec les fonctions de chaînes comme str_replace, strcmp,... Ce qui donne à cette méthode encore plus de souplesse.

Il est important de respecter la règle suivante: un minimum d'HTML dans le php et un minimum de PHP dans l'HTML. Il faudra alors utiliser beaucoup de variables avant d'utiliser les templates, il faudra aussi résister à la tentation de mettre de l'html dans la variable, ce qui n'est pas toujours possible.

| | |
|---|---|
| **ATTENTION** | les variables sont définies en-dehors du fichier template. Les IDE averissent que ces variables ne sont définies. |

## 1.6. La documentation

J'utilise Doxygen https://www.doxygen.nl/ afin de générer le code ce sont les TAGS brief, param... Cela me permet de générer une documentation; cette documentation me permet de développer plus vite et de mieux vérifier la qualité du code.

Voir le résultat ici https://wiki.noalyss.eu/doc/classes.html

## 1.7. Fichiers importants

Les fichiers à inclure sont dans la table menu_ref, cette table est accèdée à travers la table

profile_menu qui décrit l'interface de l'utilisateur, ce qui automatiquement empêche un utilisateur d'accèder à un menu auquel il n'a pas accès.

Le répertoire include contient les fichiers à inclure, les noms de fichiers suivent la logique suivant :

**ajax**
: Action ajax, le nom de fichier commence par ajax, contient le contenu de la variable op, c'est cette variable qui va indiquer le fichier à include.:footnote:voir https://wiki.noalyss.eu/doc/ajax__misc_8php.html

**fichier à inclure**
: le nom de fichier termine par inc.php,

**fichier ORM**
: correspondant à une table NOM_TABLE dans la base de données:class_NOM_TABLE_sql.php,

**fichier d'object**
: le nom commençent par class

Dans le répertoire template, vous avez les modèles, ce sont des fichiers contenant surtout du code HTML.

En résumé :

Les contrôleurs sont do.php, export.php

Les modèles sont dans include/template

et les vues sont dans le répertore include

# 1.8. La partie base de données

## 1.8.1. Les mises à jour

Depuis 2005, dans NOALYSS, il y a une table `version` qui contient les informations sur les patch de base de données qui ont été appliqués. Quand le fichier do.php est exécuté, il vérifie si **DBVERSION** contenu dans constant.php correspond à la version actuelle.

```
/* Ficher do.php */
if (DBVERSION < dossier::get_version($cn))
{
    echo '<h2 class="error" style="font-size:12px">' .
     _("Attention: la version de base de donnée est supérieure à la version du
programme, vous devriez
 mettre à jour") . '</h2>';
}
if (DBVERSION > dossier::get_version($cn))
{
    echo '<h2 class="error" style="font-size:12px">' . _("Votre base de données n'est
```

```
pas à jour") . '   ';
    $a = _("cliquez ici pour appliquer le patch");
    $base = dirname($_SERVER['REQUEST_URI']) . '/admin/setup.php';
    echo '<a hreF="' . $base . '">' . $a . '</a></h2>';
}
```

Dans le cas où **DBVERSION** n'est pas la valeur contenue dans la base de donnée, do.php vous propose d'appliquer le patch sur vos bases de données. Voici la fonction qui applique les patch SQL pour la base de données. Ces fichiers sont dans `noalyss/include/sql/patch`.

```
/* fichier setup.php */
//---------------------------------------------------------------------
// Upgrade the folders
//---------------------------------------------------------------------

for ($e=0;$e < $MaxDossier;$e++) {
   $db_row=Database::fetch_array($Resdossier,$e);
  echo "<h3>Patching ".$db_row['dos_name'].'</h3>';

  $name=$cn->format_name($db_row['dos_id'],'dos');

  if ( $cn->exist_database($name)> 0 )
  {
    $db=new Database($db_row['dos_id'],'dos');
    $db->apply_patch($db_row['dos_name']);
    Dossier::synchro_admin($db_row['dos_id']);

  } else
  {
      echo_warning(_("Dossier inexistant")." $name");
  }
 }
```

La fonction qui applique le patch est `Database::apply_patch` voir https://wiki.noalyss.eu/doc/class_database.html

Tous les patchs pour la base de données se trouve dans `/noalyss/include/sql/patch/` et sont numérotés dans l'ordre d'exécution. Chaque patch commence par

```
begin;
```

et termine par

```
insert into version (val,v_description) values (..version.,'.description..');

commit;
```

Ainsi, si le script échoue, la mise à jour s'arrête et annule toute ce qui a été fait dans le script contenant l'erreur, la base de données reste ainsi dans un état cohérent. Une fois l'erreur corrigée dans votre base de données, ce script ainsi que ceux qui suivent seront appliqués quand vous appelerez à nouveau noalyss/admin/setup.php (ou noalyss/html/install.php).

Le nombre dans la table `version` est le nombre du script moins un, exemple : le script upgrade199.sql, correspond à la 200 dans la table `version`

```
/* upgrade199.sql */

insert into version (val,v_description) values (200,'Widget and improve menu');
```

### 1.8.2. Accèder aux données : ORM

Pour accèder aux tables ou aux vues, pour les SQL complexes , nous créons une classe qui va hériter de https://wiki.noalyss.eu/doc/class_table___data___s_q_l.html qui est dérivée de https://wiki.noalyss.eu/doc/class_data___s_q_l.html

```
 et qui
va nous permettre de faire la même chose en très peu de ligne de code.
```

Cette classe fournit entre-autres les fonctions suivantes qui peuvent être surchargée

**insert**
: pour insérer une ligne dans la base de données

**delete**
: pour insérer une ligne dans la base de données

**update**
: pour mettre à jour une ligne dans la base de données

**verify**
: à surcharger, vérifie que les données sont conformes

**setp**
: setter paramètre : nom de la colonne, valeur retourne l'objet

**getp**
: getter paramètre : nom de la colonne , retourne la valeur de cette colonne

**toString**
: affiche le contenu de l'objet

**from_array**
: transforme un tableau en un objet

**seek**

recherche dans la table sur base d'une condition

**next**

retourne l'objet suivant (après seek)

**check**

contrôle des valeurs

Voici tout le code à taper par table, exemple pour la table stock_change.

Le fichier sera nommé **table**_sql.class.php , il sera dans un répertoire `/database/` et la classe se nommera **table**_SQL

```
class Stock_Change_SQL extends Noalyss_SQL
{
    // Le contructeur obligatoire
    function __construct($p_id = -1)
    {
        // Façon dont les dates sont utilisées
        $this->date_format="DD.MM.YYYY";
        // Nom de la table
        $this->table = "public.stock_change";
        // nom de la clef primaire
        $this->primary_key = "c_id";

        // Structure de la table, à gauche le nom logique utilisable
        // avec les getters/setters (setp/getp) et à droite le nom de la
        // colonne
        $this->name = array(
            "id" => "c_id",
            "c_comment" => "c_comment",
            "c_date" => "c_date",
            "tech_date"=>"tech_date",
            "tech_user"=>"tech_user",
            "r_id"=>"r_id"
        );
        // Type de données
        $this->type = array(
            "c_id" => "numeric",
            "c_comment" => "text",
            "c_date" => "date",
            "tech_date"=>"date",
            "tech_user"=>"text",
            "r_id"=>"numeric"
        );
        // Les colonnes qui ne peuvent pas être changée ni par insert ni par
        // update parce leurs valeurs sont données automatiquement
        // exemple : la clef primaire qui est un numéro de séquence
        //automatiquement donné
```

```
        $this->default = array(
            "c_id" => "auto",
            "tech_date" => "auto"
        );
        global $cn;

        parent::__construct($cn, $p_id);
    }
}
```

### 1.8.3. Les tests unitaires

**test-me**

Les fonctions info et test_me qui sont très utiles lors des phases de déboggages. Je crée une simple page test-classe.php et j'appelle la classe; dans la fonction test_me (de la classe) je mets tout ce que je souhaite tester.

exemple :

```
    static function test_me()
    {
        $cn=new Database(dossier::id());
        $obj=new Periode($cn);
        $obj->set_jrn(1);
        $obj->display_form_periode();
    }
```

| NOTE | ces tests ne sont pas ajoutés à NOALYSS mais servent de base pour les scénarios ou PHPUNIT |
|---|---|

**PHPUNIT**

Il faut tout d'abord un fichier bootstrap qui contient le path correct pour php ainsi que les variables comme $g_user ou $cn, l'inclusion de `noalyss/include/config.inc.php` et de `noalyss/include/constant.php`

**Scenario**

Vous pouvez enregistrer ce que vous soumettez, il faut dans `config.inc.php`, ajouter

```
   define ('LOGINPUT',true);
```

Ensuite créer le fichier dans `noalyss/html/authorized_debug` , le contenu n'a pas d'importance, ce fichier peut être vide.

Ensuite, aller dans votre dossier de test et faites une action (une vente, un achat...), vous devez

ouvrir le fichier test.php avec comme paramètre le dossier (exemple http://localhost/noalyss/html/test.php?gDossier=14) ce que vous venez de faire a été sauvé dans le répertoire défini par `$_ENV['TMP']`, (sous linux il s'agit de /tmp )avec un nom ressemblant à scenario-<nombre>.php

Vous devez d'abord copier ce fichier dans le répertoire noalyss/scenario , de préférence avec un nom plus parlant.

Si vous pointez votre browser sur noalyss/html/test.`php (après avoir créé le fichier authorized_debug) en cliquant sur le lien avec le nom de fichier vous pourrez rejouer l'action.

| | |
|---|---|
| **ASTUCE** | Vous pouvez améliorer la description en changeant l'annotation `//@description: <CODE>` |

L'objectif étant de pouvoir tester et de rejouer facilement les actions que vous avez faites. Cela permet de débogguer plus facilement en particulier les parties ajax.

[1] Pour l'ajax, il existe plusieurs fichiers, ajax_misc.php est en général utilisé,ajax.php est utilisé pour les pluging

# Chapitre 2. Développement de widget

## 2.1. Définition

Un widget est un élément sur le tableau de bord qui affiche le suivi, des rapports, un agenda,...

## 2.2. Développement

le nom code du widget , est toujours le même que celui du sous-répertoire dans noalyss/include/widget, que le nom du fichier principal et aussi, le nom de la CLASS.

| | |
|---|---|
| **IMPORTANT** | ce nom ne peut pas contenir d'autres caractères que des lettres et des soulignés, de préférence toujours en minuscule. |

Par exemple: le widget avec le code event (voir table widget_dashboard) est le nom du sous-répertoire de noalyss/include/widget

| | |
|---|---|
| **NOTE** | Les widgets doivent fonctionner dans le namespace `Noalyss/Widget` |

## 2.3. Installation

Création d'un fichier appelé "install.php" dans le sous-répertoire noalyss/include/widget qui sera exécuté si le widget n'est pas encore dans la base de données. Quand l'utilisateur est sur le tableau de bord, qu'il clique sur "Personnaliser" puis "Ajouter", le programme va scanner le répertoire noalyss/include/widget à la recherche de fichiers install.php, il va vérifier que chaque fichier est dans la base de données ou l'insère s'il n'y est pas.

Le code du fichier install.php est simplement d'insérer le widget dans la base de données et éventuellement peut initialiser d'autres choses.

**structure table WIDGET_DASHBOARD**

Contient tous les widgets ,

| Colonne | Type de données | Description |
|---|---|---|
| wd_id | int | clef primaire (valeur automatique) |
| wd_code | texte | code du widget (doit être unique , pas de souligné ou de caractères interdit, ce sera aussi le nom de la classe ) |
| wd_description | texte | description du widget , ce qu'il fait |
| wd_parameter | int | égal à 0, s'il n'y a pas de paramètre, à 1 s'il existe un paramètre (voir Paramètre) |

Exemple

```
global $cn;
```

```
$cn->exec_sql("insert into widget_dashboard
(wd_code,wd_description,wd_parameter,wd_name) values ($1,$2,$3,$4)",
array (
    'event'
    ,'Affiche les 10 actions en retards, celles à venir , les 10 prochaines factures
client ou fournisseurs, ou celles en retard '
    ,1
    ,'10 actions ou factures'
    )
);
```

## 2.4. Ajax

Tous les appels Ajax appellent "html/ajax_misc.php" , la requête doit contenir :

- la variable **op** : "widget"
- la variable **w** : chaîne qui est le code du widget (nom du sous-répertoire de include/widget)

le fichier Ajax.php du sous-répertoire NOALYSS_INCLUDE/widget/(w)/Ajax.php sera appelé

## 2.5. Application

La classe s'appelle toujours "nom-du-widget.php" , elle est dérivée de widget et doit avoir les fonctions

- display : affichage du widget
- input : affichage de la description et permet son activation (visible dans la box )
- input-parameter : si des paramètres doivent être sauvées, les paramètres sont par utilisateur et par widget activés,
- display_parameter

### 2.5.1. Paramètres

Certains widgets peuvent être paramétrés, et il est possible d'ajouter plusieurs fois le même widget

Exemple mini-report : choix du mini-report à afficher,

Pour cela, il faut ajouter certaines fonctions :

**input_parameter() : création du FORM**

création d'un FORM dont le DOMID sera le code widget (wd_code) suivi de "_param"

exemple pour mini_report, on utilise la fonction `Widget->make_form` , qui enveloppe le code transmis dans une balise FORM, construisant ainsi le code HTML d'un FORM qui sera envoyé correctement en Ajax. Les paramètres passés seront sauvés dans `user_widget` et pourront être récupérés dans un

tableau associatif avec `Widget→get_parameter`.

```
    function input_parameter() {

        $select=new \ISelect('simple_report');
        $select->value=$this->db->make_array("select fr_id, fr_label from
form_definition order by 2");
        $this->make_form($select->input());
    }
```

**display_parameter : affichage des paramètres**

Les paramètres sont toujours sauvés "brut" , comme une chaîne URL par la fonction `Widget→make_form()`, il faut donc pouvoir l'afficher, en la travaillant avec `parse_str`, pour cela il faut appeler `Widget→get_parameter()`, le résultat doit être dans un SPAN avec la classe "widget_param".

Exemple pour mini_report

```
    function display_parameter() {
        $aParam=$this->get_parameter();①
        $name = $this->db->get_value("select fr_label from form_definition where
fr_id=$1",[$aParam['simple_report']]);
        echo " ";
        echo span(_("Rapport") ." ".h($name),' class="widget_param"');


    }
```

(1) renvoie les paramètres dans un tableau associatif

### 2.5.2. function  display

Toujours commencer par un DIV (id=code_widget+uw_id), Widget::open_div et Widget::close_div

Puis avoir un titre et un bouton "Agrandissement", avec la fonction Widget::button_zoom ou utiliser la fonction `Widget→title()`

*Example*

```
echo HtmlInput::title_box('titre du widget', uniqid(), 'custom', $this->button_zoom(),
'n');

// ou plus simplement

echo $this->title('Titre du widget');
```

Ce DIV doit avoir comme classe **box** et **widget-box** , le **widget-box** permet de numéroter les boites.

### 2.5.3. Exemple complet

| ASTUCE | Toute la documentation du code source est maintenue avec Doxygen, ici on a un exemple. |
|---|---|

```
/*!
 * \file
 * \brief display the next invoice to be paid or late for customer or supplier
 */
namespace Noalyss\Widget;
/*!
 * \class
 * \brief display the next invoice to be paid or late for customer or supplier
 */
class Invoice extends Widget
{
    /**
     * @brief return the constant array Tiers
     * @return array
     */
    static function getConstantTiers() : array
    {
        return ['S' => _("Fournisseurs"), "C" => _("Clients")];;
    }
    /**
     * @brief return the constant array Limit
     * @return array
     */
    static function getConstantLimit() :array {
        return  ['P' => _("Prochaines factures"), "R" => "facture en
retard",'T'=>_("Aujourd'hui")];
    }

    /**
     * @brief let choice what to display
     * @return void
     */
    function input_parameter()
    {
        $tiers = new \ISelect('tiers');
        $aTiers=Invoice::getConstantTiers();
        $tiers->value=[];
        foreach ($aTiers as $key=>$value) {
            $tiers->value[]=['value'=>$key,'label'=>$value];
        }
        $time_limit = new \ISelect('time_limit');
        $aLimit=Invoice::getConstantLimit();
        $time_limit->value=[];
        foreach ($aLimit as $key=>$value) {
            $time_limit->value[]=['value'=>$key,'label'=>$value];
```

```
        }

        $input = _("Factures ") . $tiers->input() . " " . _("échéance") . " " .
$time_limit->input();
        $this->make_form($input);

    }

    /**
     * @brief display the parameter
     * @return void
     */
    function display_parameter()
    {
        $aParam = $this->get_parameter();
        $aTiers =Invoice::getConstantTiers();
        $aLimit = Invoice::getConstantLimit();
        echo '<span class="widget_param">'.$aTiers[$aParam['tiers']] . " " .
$aLimit[$aParam["time_limit"]].'</span>';
    }

    /**
     * @brief display the widget
     * @return void
     * @throws \Exception
     */
    function display()
    {
        $this->open_div();
        $aParam = $this->get_parameter();
        $aTiers = Invoice::getConstantTiers();
        $aLimit = Invoice::getConstantLimit();
        $title = $aTiers[$aParam['tiers']] . " " . $aLimit[$aParam["time_limit"]];

        $this->title($title);

        $acc_ledger = new \Acc_Ledger($this->db, 0);

        $ledger_type = 'ACH';
        if ($aParam['tiers'] == 'C') {
            $ledger_type = 'VEN';
        }

        switch ($aParam['time_limit']) {
            case 'P':
                $array = $acc_ledger->get_operation_date(date('d.m.Y'), $ledger_type,
'>');
                break;
            case 'R':
                $array = $acc_ledger->get_operation_date(date('d.m.Y'), $ledger_type,
'<');
```

```
                break;
            case 'T':
                $array = $acc_ledger->get_operation_date(date('d.m.Y'), $ledger_type,
'=');
                break;
        }
        include "invoice-display.php";
        $this->close_div();


    }
```

Voir aussi "Paramètres"

# Chapitre 3. Développement d'extension (plugin)

## 3.1. L'API

Toute la documentation se trouve sur http://wiki.noalyss.eu/doc/; seules les classes nous intéressent. La documentation est générée à partir du code avec Doxygen. Le but de cet article n'est pas de revisiter la documentation technique mais de la mettre en oeuvre afin d'écrire une extension ou plugin.

## 3.2. Installation de l'extension

Les extensions se trouvent toujours dans un sous-répertoire de `/noalyss/include/ext`, pour notre plugin que nous appelerons DUMMY, le répertoire correspondant sera `/noalyss/include/ext/dummy`. La première étape est donc de décompresser le plugin dans le répertoire `/noalyss/include` et vérifier si on a bien `/noalyss/include/ext/DUMMY` ensuite l'activer.

Le plus simple est de retrouver le fichier dummy.php, si le chemin est `/chemin/ext/dummy/dummy.php` alors /chemin/ext est l'endroit où les plugins doivent être décompressés. Le chemin quand on ajoute le plugin dans le menu CFGMENU est donné dans le wiki , pour chaque plugin, on donne le paramètre "chemin" qui est en fait le chemin à partir de /ext/

## 3.3. Comment l'activer

Grâce à l'accès direct allez sur CFGMENU, puis ajoutez plugin, les champs sont les mêmes, il faut ajouter ce plugin dans le profil des utilisateurs qui doivent l'utiliser (CFGPRO)

Pour la version 6.8 et plus haut, il suffit d'aller dans CFGPLUGIN et cochez qui peut utiliser le plugin qui se trouvera dans le module "Extension". Avec CFGMENU, on peut placer l'extension dans un autre module.

### 3.3.1. Explication des champs

- Label, est le nom de menu de votre extension
- Code est utilisé pour inclure le fichier du plugin, il correspond au champs caché plugin_code
- Fichier est le chemin complet vers l'extension

Pour notre extension, les valeurs suivantes sont données

- Label : Mon dummy à moi
- code : dum
- Fichier : dummy/dummy.php

Vous sauvez, cliquer sur Extension et vous verrez apparaître un nouveau plugin appelé "Mon dummy à moi"; si vous cliquez dessus, la page ext/dummy/dummy.php sera exécutée; voyez le code

du fichier noalyss/html/extension.php pour comprendre.

### 3.3.2. Tutoriel vidéo

Installation d'un plugin sur Noalyss

### 3.3.3. Exemple de base

Dans les plugins de base de NOALYSS, il y a skel qui est le squelette d'un plugin. Vous pouvez l'utiliser comme base.

https://gitlab.com/noalyss/noalyss-plugins/-/tree/stable/skel?ref_type=heads

### 3.3.4. Connection à la base de données

Certaines valeurs doivent toujours être passées à chaque page, par exemple gDossier qui est l'identifiant du dossier. Pour se connecter c'est assez facile, il faut utiliser la classe Database et la classe Dossier.

Donc on écrit dans dummy.php

```
 echo "L' identifiant de mon dossier est ".dossier::id()."<br>";
 echo "Son nom réel est ".DOMAIN."dossier".dossier::id()."<br>";
 echo "Son nom est ".dossier::name()."<br>";
// Je me connecte à présent à ce dossier
 $cn_db=new Database(dossier::id());
 // On peut aussi se connecter ainsi (développement plus récent)
 $cn= Dossier::connect();
```

Je veux afficher toutes les fiches qui concernent le matériel à amortir, donc j'ai besoin de savoir ce que vaut son FICHE_DEF_REF:FRD_ID dans constant.php; la FICHE_TYPE_XX n'existe pas, je vois dans la table fiche_def_ref qu'il s'agit de "7 Matériel à amortir"

Pour avoir toutes les fiches,

```
$fiche= new Fiche($cn_db);
$aFicheMateriel=$fiche->getByDef(7);
```

Pour chaque fiche, je veux afficher son nom, son prix, le nombre d'années à amortir et la date d'achat. Je peux en trouver les valeurs dans la table attr_def ou le fichier constant.php,

Donc cela devient

```
for ($i=0; $i < count($aFicheMateriel);$i++) {
  echo "<ul>
   echo "<li> ";
   echo "Nom";
   echo $aFicheMateriel[$i]->strAttribut(ATTR_DEF_NAME);
```

```
    echo "</li>";
    echo "<li> ";
    echo "Prix achat";
    echo $aFicheMateriel[$i]->strAttribut(ATTR_DEF_PRIX_ACHAT);
    echo "</li>";
    echo "<li> ";
    echo "Durée amortissement";
    echo $aFicheMateriel[$i]->strAttribut(8);
    echo "</li>";
    echo "<li> ";
    echo "Date de début";
    echo $aFicheMateriel[$i]->strAttribut(10);
    echo "</li>";
    echo "</ul>
  }
```

### 3.3.5. Soumettre des requêtes

Uniquement pour l'exercice, nous allons ajouter un FORM. Les données nécessaires dans le FORM sont toujours au minimum: l'id du dossier, le code du plugin.

Dans le FORM, on demandera juste à afficher le solde de chaque élément. On aura alors le code suivant, on utilisera la technique des templates

```
  $year=new IText('year');

  $str_year=$year->input();

  $str_submit=HtmlInput::submit('year_left','Appliquer');

  require_once('template1.php');
```

```
template1.php
```

```
  <FORM METHOD="GET" ACTION="extension.php">

  <?=dossier::hidden()?>

  <?=HtmlInput::extension()?>

  Solde pour l'année : <?=$str_year?>

  <?=$str_submit?>

  </form>
```

Puis dans le début du fichier noalyss/include/ext/dummy/dummy.php, on ajoutera un test pour savoir un FORM a été soumis et on affichera une boîte de dialogue.

```
$http= new HttpInput();①
 if (isset($http->get('year_left','number'))){ ②

    alert('Vous avez demandé le nombre d\'années restantes');

 }
```

(1) permet de retrouver des variables passées par get , post ou request (voir documentation code^)
(2) remplace `$_GET['year_left']`, HttpInput vérifie aussi le type de données et peut donner une valeur par défaut si cette donnée n'est pas dans $_GET

## 3.4. Plugin plus avancé

Ma première extension, intégrer un fichier de client dans une catégorie de fiche, ce fichier est en CSV. Le code est simple et compréhensible, normalement on devrait avoir une meilleure gestion des erreurs, vérifier les attaques SQL Inject,... Ce code n'est là QUE pour expliquer le concept. On n'a pas utilisé plusieurs pages, ni de templates

Tout d'abord, il faut se connecter à la base de données

```
 // se connecter au dossier courant

 $cn=Dossier::connect();
```

Dans extension.php on vérifie la sécurité, en ajoutez une dans l'extension n'est en général pas nécessaire mais vous pourriez avoir votre propre système de sécurité si votre extension est fort complexe

En premier lieu, il est nécessaire de choisir dans quelle catégorie de fiche je veux intégrer les enregistrements. Donc on utilise un petit form

```
 echo '<form METHOD="get" action="extension.php">';

 echo dossier::hidden();

 // Ceci vous permet de revenir ici (voir extension.php). Cet élément caché permet
d'include cette page-ci

 // Donc si votre plugin contient plusieurs pages, vous allez devoir ajouter une
seconde variable pour

 // inclure la page que vous voulez (voir méthode de développement de PhpCompta )

 echo HtmlInput::extension();
```

```
 echo "Choix de la catégorie de fiche";

 $select_cat=new ISelect('fd_id');

 $select_cat->value=$cn->make_array('select fd_id,fd_label from fiche_def where
frd_id='.

     FICHE_TYPE_CLIENT);

 echo $select_cat->input();

 echo HtmlInput::submit('display_prop','Afficher les propriétés');


 echo '</FORM>';
```

Il faut remarquer 2 choses dans ce FORM, primo, on utilise les objets HtmlInput et ISelect, secundo on doit avoir absolument en variables cachées, le n° de dossier sur lequel on est connecté, le code de l'extension, qui permettra à extension.php d'include le bon fichier. On utilise ici le protocole GET puisqu'on interroge, le protocole POST est réservé aux sauvegardes, c'est une convention assez répandue. La différence, est que les requêtes GET se voient dans l'URL, les requêtes POST ne sont jamais dans l'url.

L'utilisateur soumet le FORM, donc la feuille se recharge et on arrive à cette partie du code

On choisit d'afficher les propriétés avant de confirmer l'import

```
 if ( isset($_GET['display_prop'])){
    $http=new HttpInput();
     $a=new Fiche($cn);

     $prop=$a->toArray($http->get('fd_id'));

     foreach ($prop as $key=>$value)     echo "Index : $key valeur $value <br/>";



     echo '<form method="POST" action="extension.php"  enctype="multipart/form-
data">';

     echo dossier::hidden();

     echo HtmlInput::extension();

     echo HtmlInput::hidden('fd_id',$http->get('fd_id');

     $file=new IFile('fichier_csv');
```

```
     echo $file->input();

     echo HtmlInput::submit('start_import','Démarrez importation');

     echo '</form>';

     exit;

 }
```

Voilà, si l'utilisateur clique sur le bouton SUBMIT, l'importation va démarrer. Dans notre exemple, on imaginera que le fichier CSV n'a que 4 champs "nom client","prenom client", "numero client","adresse client"

Le code qui suit est très simplifié.

```
 if ( isset($_POST['start_import'])){
$http=new HttpInput();
     $fd_id=$http->post('fd_id');

     $tmp_file=$_FILE['fichier_csv']['tmp_name'];

     if ( ! is_uploaded_file($tmp_file))

         die 'Je ne peux charger ce fichier';

     // on ouvre le fichier

     $f=fopen($tmp_file,'r');

     // On récupère les propriétés de cette catégorie de fiche

     $client=new Fiche($cn);

     // $array contient toutes les valeurs nécessaires à Fiche::insert,

     $array=$client->toArray($http->post('fd_id'));



     while ( $data=fgetcsv($f)) {

         // remarque : on a éliminé les traitements d'erreur



         // On  remet tous les attributs (propriétés) à vide

         foreach(array_keys($array) as $key) $array[$key]="";
```

```
        // Nom et prénom
        $array['av_text1']=$data[0].' '.$data[1];
        // Numéro de client
        $array['av_text30']=$data[2];
        // Adresse
        $array['av_text14']=$data[3];
        // Quickcode
        $array['av_text23']="CLI".$data[2];
        $client->insert($fd_id,$array);
    }
    exit;
}
```

Voici le fichier client.txt

```
"Nom client1","Prénom","C1","Rue de la boite,55"
"Nom client2","Prénom","C2","Rue du couvercle,55"
"Nom client3","Prénom","C3","Rue de la chaussure,55"
"Nom client4","Prénom","C4","Rue de la couleur,55"
```

Si vous vérifiez dans VW_CLIENT, vous verrez que toutes vos fiches ont été ajoutées. Dans l'exemple, il fatraitement d'erreur plus élaboré; le fait que si une fiche echoue , l'opération est annulée (Database::rollback) ou alors création d'un fichier avec les enregistrements "ratés"...

## 3.5. Ajax

Afin d'utiliser des fonctions avec ajax, prototype.js devrait être utilisé.Vous devez appeler le fichier ajax.php, ce fichier dans html va simplement vérfier la sécurité et appeler le fichier ajax.php du répertoire où se trouve le plugin avec tous les arguments donnés.

Exemple

dans

dummy/javascript.js, vous avez

```
function show_detail(pop_id){
    $('detail_invoice_content').innerHTML=loading();
    showIPopup('detail_invoice');
    try {
    var gDossier=$('gDossier').value;
    var phpsessid=$('phpsessid').value;
    var code=$('code').value;
    var
obj={"op_id":pop_id,"gDossier":gDossier,"phpsessid":phpsessid,"code":code,'act':'detai
l_invoice'};
    var queryString=encodeJSON(obj);
    var action=new Ajax.Request ( 'ajax.php',
              {
                 method:'get',
                 parameters:queryString,
                 onFailure:show_detail_error,
                 onSuccess:show_detail_success
               }
               );
    } catch (e){alert('show_detail'+e.message);}


 }
```

et dans dummy/ajax.php

```
<?php
// Met correctement la langue

set_language();
//retrouve le dossier courant et s y connecte

$gDossier=dossier::id();
$cn=new Database($gDossier);

// action

$action=(isset($_REQUEST['act']))?$_REQUEST['act']:'sh';


// Véridfie la sécurité

require_once ('class_user.php');
$User=new User(new Database());
$User->Check();
```

```
/* Suivant l action demandé, on executera tel ou tel partie de code

/* Show the document */
if ( $action == 'sh') {
/*

* Votre code

*/}
/* remove the document */
if ( $action == 'rm' ) {
/*votre code et la réponse

*/

    header('Content-type: text/xml; charset=UTF-8');
    header ('<?xml version="1.0" encoding="UTF-8"?>');
    echo '<answer>';
    echo '<ctl>detail_invoice</ctl>';
    echo '<html>'.escape_xml($html).'</html>';
    echo '</answer>';
}
?>
```

## 3.6. Les données

Si votre extension nécessite de sauver ses propres données, il faut impérativement les mettres dans un schéma séparé. Prévoyez une table version, afin d'appliquer des mises à jour si nécessaire et un fichier install.php afin de créer les schémas nécessaire

exemple

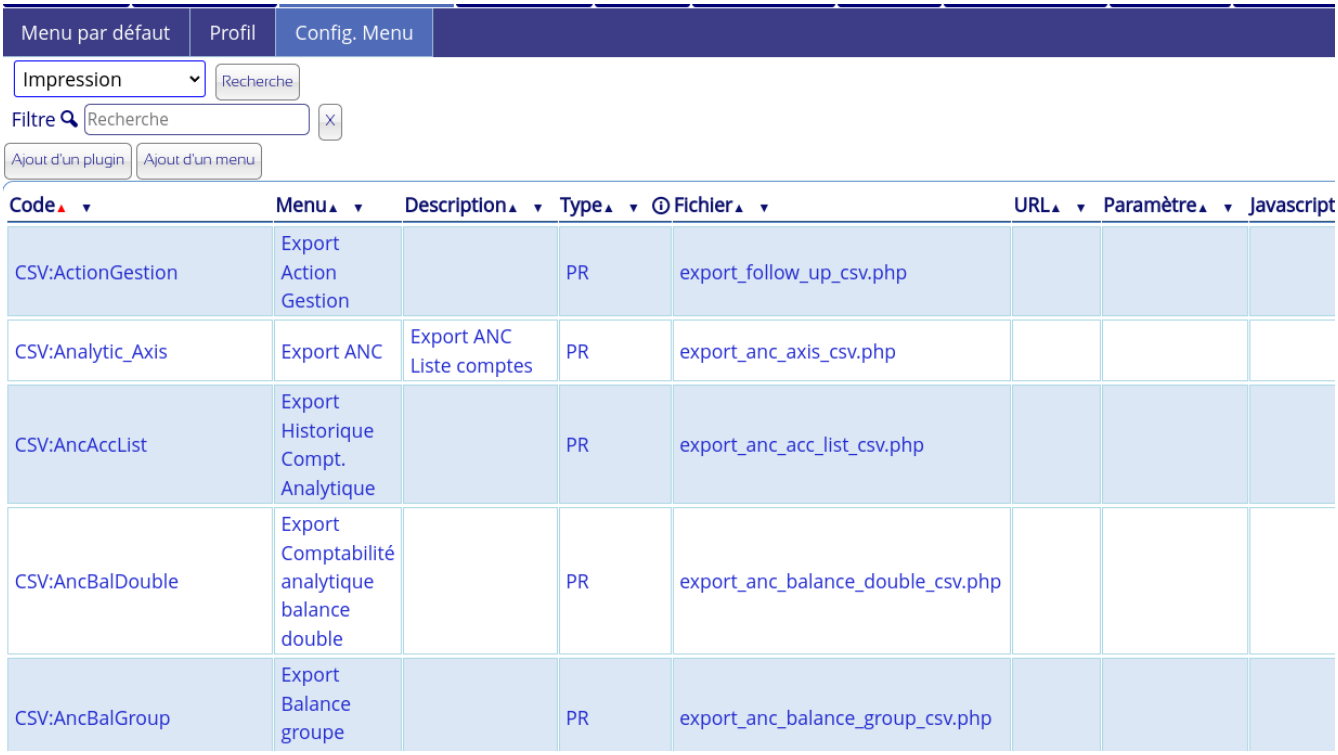
```
create schema plugin_tva;

create table version (val integer);
```

## 3.7. Comment faire un rapport en PDF

Si vous décidez d'ajouter un menu qui renvoit un PDF.

Tout d'abord comme dans le développement des plugins, il faut d'abord ajouter un menu dans les menu et les [menu/cfgpro[profils.

Les menus de type impression ne sont jamais affichés, ils ne servent qu'à la sécurité et commence toujours par `CSV:`,`PDF:`,... voir [C1MENU]




| Code | Menu | Description | Type | Fichier | URL | Paramètre | Javascript |
|---|---|---|---|---|---|---|---|
| CSV:ActionGestion | Export Action Gestion | | PR | export_follow_up_csv.php | | | |
| CSV:Analytic_Axis | Export ANC | Export ANC Liste comptes | PR | export_anc_axis_csv.php | | | |
| CSV:AncAccList | Export Historique Compt. Analytique | | PR | export_anc_acc_list_csv.php | | | |
| CSV:AncBalDouble | Export Comptabilité analytique balance double | | PR | export_anc_balance_double_csv.php | | | |
| CSV:AncBalGroup | Export Balance groupe | | PR | export_anc_balance_group_csv.php | | | |

Le code , vous devez créér une nouvelle classe depuis la classe PDF (en mode portrait) ou la classe PDFLand en mode paysage

Exemple

```
require_once('class_pdf.php');

class Mon_Impression_PDF extends PDF
{

    function header()
    {
        parent::header();

    }

    function export()
    {

    }
}
```

La fonction header : est ce qui est affiché comme en tête sur chaque page, La fonction footer : est ce qui est affiché comme en pied de page sur chaque page,

La fonction la plus importante est export qui remplira le listing.

Il y a peu de fonctions à connaître : Cell, export , SetFillColor…

Voici par exemple le morceau de code pour le listing dans amortissements

```
class Amortissement_Material_PDF extends PDF
{

    function header()
    {
        parent::header();
        $this->setFont('DejaVu', 'B', 14);
        $this->Cell(190, 10, _('Amortissement : Liste de biens'), 1, 2, 'C');
        $this->ln();
        $this->col_size=array('qcode'=>20, 'name'=>35,'desc'=>80, 'date.purch'=>20,
'year.purch'=>20,  '#amort'=>10,'amount.purch'=>30, 'amount.amort'=>30, '%'=>20,
'amount.remain'=>20,'amount.delta'=>20);

        $this->setFont('DejaVu', 'B', 7);
        $this->Cell($this->col_size['qcode'], 8, _('QCode'));
        $this->Cell($this->col_size['name'], 8, _('Nom'));
        $this->Cell($this->col_size['date.purch'], 8, _('Date achat'));
        $this->Cell($this->col_size['year.purch'], 8, _('Année achat'));
        $this->Cell($this->col_size['#amort'], 8, _('Nbre'), 0, 0, 'R');
        $this->Cell($this->col_size['amount.purch'], 8, _('Montant'), 0, 0, 'R');
        $this->Cell($this->col_size['amount.amort'], 8, _('A amortir'), 0, 0, 'R');
        $this->Cell($this->col_size['amount.delta'], 8, _('A amortir'), 0, 0, 'R');
        $this->Ln();
    }

    function export()
    {
        global  $cn;
        $this->SetFont('DejaVu', '', 7);
        $ret=$cn->get_array("select * from amortissement.v_amortissement_summary where
a_visible='Y' order by a_start,a_date");
        bcscale(2);
        for ($i=0;$i<count($ret);$i++)
        {
            if ($i%2==0)
            {
                $this->SetFillColor(220, 221, 255);
                $fill=1;
            }
            else
            {
                $this->SetFillColor(0, 0, 0);
                $fill=0;
            }
            $this->Cell($this->col_size['qcode'], 8, $ret[$i]['quick_code'], 0, 0,
'L', $fill);
            $this->Cell($this->col_size['name'], 8, $ret[$i]['vw_name'], 0, 0, 'L',
$fill);
            $this->Cell($this->col_size['date.purch'], 8,
format_date($ret[$i]['a_date']), 0, 0, 'L', $fill);
```

```
            $this->Cell($this->col_size['year.purch'], 8, $ret[$i]['a_start'], 0, 0,
'C', $fill);
            $this->Cell($this->col_size['#amort'], 8, round($ret[$i]['a_nb_year']), 0,
0, 'R', $fill);
            $this->Cell($this->col_size['amount.purch'], 8, nb($ret[$i]['a_amount']),
0, 0, 'R', $fill);
            $this->Cell($this->col_size['amount.amort'], 8,
nb($ret[$i]['amort_done']), 0, 0, 'R', $fill);
            $delta=bcsub($ret[$i]['a_amount'],$ret[$i]['amort_done']);
            $this->Cell($this->col_size['amount.delta'], 8, nb($delta), 0, 0, 'R',
$fill);

        $this->Ln();
        }

        $this->Output('listing-amort.pdf', 'I');
    }

}
```

Il est conseillé de travailler avec des vues plutôt qu'avec des requêtes SQL directes, plus facile de réutiliser une vue que de dupliquer à plusieurs endroits une requête SQL, surtout pour la maintenance si vous décidez de changer la requête ou la structure de la base de données, avec une vue, vous retrouverez plus vite les endroits à modifier.

Il y a des tutoriaux dans noalyss/doc/fpdf et tfpdf, vous pouvez les lire en ligne sur le site de fpdf