Principes de développement et/ou de portage des modules sous NPDS

Rappel :

- Un module doit être considéré comme un composant additionnel à la plate-forme de base.

Principes :

- Un module ne peut pas (et ne doit pas) intervenir dans les fichiers du noyau
- Un module peut intervenir dans la DataBase

- Un module est hébergé dans une structure particulière de l'organisation du noyau, à savoir:

  ⇨ Mon_web/**modules**/mon_module

  ⇨ La totalité du module dans ce trouver localisé dans ce sous-répertoire

  ⇨ Dans le cas d'un module disposant de son propre sous-ensemble d'administration, de page d'aide …, la structure doit être :

  ⇨ mon_module/admin
  ⇨ mon_module/manuel
  ⇨ …

- Un module doit comporter au **minimum** :

  ⇨ un fichier de licence (GNU /GPL par exemple)
  ⇨ mon_module.php
  ⇨ un fichier index.html (vide)

- autant que de besoin, on trouvera :

  ⇨ mon_module.conf.php (configuration)
  ⇨ mon_module.conf.lang (langue / avec une fonction mon_module_translate)
  ⇨ mon_module.sql (modification / création de tables)

Lancement :

- Le lancement se fera par l'appel au fichier du noyau : modules.php

  ⇨ modules.php?**ModPath**=mon_module&**ModStart**=mon_module

  ⇨ Le fichier assure le lancement du module et la fourniture de deux variables php permettant la gestion des url, liens …

  $ModPath et $ModStart

Cartouche fondamental du fichier principal d'un module :

Exemple :

```
…
/* This program is free software. You can redistribute it and/or modify */
/* it under the terms of the GNU General Public License as published by */
/* the Free Software Foundation; either version 2 of the License.       */
/************************************************************************/

if(!IsSet($mainfile)) { include ("mainfile.php"); }

// Permanent double-side theme
global $pdsts;
$pdst="1";

include ("modules/$ModPath/mon_module.conf.php");
include ("modules/$ModPath/mon_module.lang.php");
```

on retrouve donc :

- le chargement (si nécessaire) de mainfile.php
- la déclaration et l'initialisation de la variable $pdsts (0= simple-Side, 1=Double-Side)
- les includes nécessaires :
  - le .conf.php
  - le .lang.php

Utilisation de SuperCache :

Un module peut (et doit autant que possible) utiliser le SuperCache de NPDS

A partir de **NPDS 4.8 P2**, une définition globale pour tous les modules existe dans le fichier cache.timings.php de la racine de NPDS.

Ce fichier implémente pour tous les modules un temps de cache commun (par défaut 3600 secondes) pour l'ensemble des URL manipulées.

```
$CACHE_TIMINGS['modules.php'] = 3600; (
$CACHE_QUERYS['modules.php'] = "^";
```

Le développeur, peut donc ne pas implémenter de fichier cache.timings.php particulier pour son module.

Cependant, s'il le souhaite, un fichier de configuration de SuperCache doit donc être copier dans l'arborescence du module

⇨ cache.timings.php (configuration de SuperCache)

```
$CACHE_TIMINGS['modules.php'] = 86400; (durée en seconde du cache)
$CACHE_QUERYS['modules.php'] = "^ModPath=mon_module&ModStart=mon_module";
```

⇨ adapter ce fichier à votre module.

⇨ attention à l'ordre de fabrication de vos URL (et notamment `ModPath=mon_module&ModStart=mon_module`), car ne seront cachées que les URLs qui seront formées dans le « bon sens » (entendez par la suite ordonnée de paramètres).

Donc dans notre exemple `ModStart=mon_module&ModPath=mon_module` ne sera pas caché.

L'implémentation du fichier cache.timings.php dans le cartouche :

```
...
/* This program is free software. You can redistribute it and/or modify */
/* it under the terms of the GNU General Public License as published by */
/* the Free Software Foundation; either version 2 of the License.       */
/************************************************************************/
if(!IsSet($mainfile)) { include ("mainfile.php"); }

// Include cache manager classe
include_once('cache.class.php');

// Permanent double-side theme
global $pdsts;
$pdst="1";

include ("modules/$ModPath/mon_module.conf.php");
include ("modules/$ModPath/mon_module.lang.php");
include ("modules/$ModPath/cache.timings.php");
```

Exemple de fonction :

```
function test() {
   // Cette ligne permet la récupération des varaibles globales;
   global $ModPath, $ModStart, $SuperCache;

   include("header.php");

   // Include cache manager
   if ($SuperCache) {
      $cache_obj = new cacheManager();
      $cache_obj->startCachingPage();
   }
   if (($cache_obj->genereting_output==1) or ($cache_obj->genereting_output==-1) or (!$SuperCache)) {
     OpenTable();
     $TableRep=mysql_query("SELECT * FROM test ORDER BY nom LIMIT $debut,$nb_affichage");
     $topsuivant="modules.php?ModPath=$ModPath&ModStart=$ModStart&op=rech_lettre&lettre=$lettre";
     $NombreEntrees=mysql_NumRows($TableRep);

     pagination ($top, $debut,$topsuivant,$nb_affichage,$nbe);
     aff_alpha("pied");
     CloseTable();
   }
   if ($SuperCache) {
      $cache_obj->endCachingPage();
   }

   include("footer.php");
}
```

Le module glossaire est un bon exemple de l'implémentation (en fait un portage) de l'architecture des modules dans NPDS (qui ressemble à celle de PostNuke et/ou de PhpNuke)

Cas particulier des modules disposant d'une administration via NPDS-Plugins (disponible à partir de NPDS 4.8 Patch 3):

Un module peut (et doit autant que possible) utiliser NPDS-Plugins pour son administration.

Plugins est un interface simple qui est basé essentiellement sur :

1 - une déclaration quasi XML du module dans un fichier particulier : extend-modules.txt
- ce fichier (extend-modules.txt) se trouve dans le répertoire admin de NPDS
- la syntaxe de déclaration :

[module]
[nom]**Bannir une IP**[/nom]
[ModPath]**ipban**[/ModPath]
[ModStart]**admin/setban**[/ModStart]
[/module]

⇨ on retrouve ModPath et ModStart qui sont directement issue de l'architecture des modules.
- Le fichier extend-modules.txt est **unique** mais peut parfaitement être modifié, complété.

2 – le script d'admin a proprement parler qui sera inclus dans l'admin via plugins.php

Le Switch qui doit être utilisé pour activer les fonctions de votre module est **$subop** (et non pas **$op** comme traditionnellement). L'$op pour Plugins est nécessairement : « Extend-Admin » ou « Extend-Admin-SubModule »

```
switch($subop) {
  case "SAVE":
    Save($ipban, $ModPath, $ModStart);
  break;

  default:
    Configure($ModPath, $ModStart);
  break;
}
```

Les <form> doivent au minimum contenir les lignes suivantes :

```
// C'est bien admin.php qui appelé mais avec les codes opérations op ET subop
echo"<form action=\"admin.php\" method=\"post\">";
    ..
    ..
// lignes obligatoires pour pouvoir utiliser NPDS-plugins
echo "<input type=\"hidden\" name=\"op\" value=\"Extend-Admin-SubModule\">";
echo "<input type=\"hidden\" name=\"ModPath\" value=\"$ModPath\">";
echo "<input type=\"hidden\" name=\"ModStart\" value=\"$ModStart\">";
// sous-opération du module
echo "<input type=\"hidden\" name=\"subop\" value=\"VOTRE_SUBOP\">";
// lignes obligatoires pour pouvoir utiliser NPDS-plugins
    ..
echo "</form>";
```

Il faut systématiquement penser à globaliser (ou transmettre dans les appels) $ModPath et $ModStart pour pouvoir assurer le fonctionnement correcte de votre module.

De ce point de vue le module IPBAN est un exemple presque parfait (et simple).