Programmation de modules postnuke

Code: portal-prog-pn

Originaux

url: http://tecfa.unige.ch/guides/tie/html/portal-prog-pn/portal-prog-pn.html
url: http://tecfa.unige.ch/guides/tie/pdf/files/portal-prog-pn.pdf

Auteurs et version

- Vivian Synteta - Daniel K. Schneider - Stéphane Morand

- Version: 0.3 (modifié le 20/6/02)

Prerequis

Module technique précédent: mysql-intro
Module technique précédent: php-mysql

Abstract

Il s'agit d'un module qui montre les principes de base pour dévélopper un module pour le portail "Postnuke"

Objectifs

- Décrire l'architecture du portail PostNuke
- Expliquer la notion d'un module
- Montrer les étapes à suivre pour créer un module "postnuke"
- Donner des conseils

1. Table des matières

2. Introduction

- Postnuke = portail communautaire, architecture assez modulaire.
- Un module c'est:
  - une application qui augmente les fonctionalités du portail et qui s'intègre facilement dans sa structure de base (ex: forum, Top10).
  - un répertoire qui contient des fichiers avec des fonctions
- Un module peut être:
  - soit indépendant (ex: forum) qui exécute des tâches indépendamment des autres tâches du portail et qui utilise le portail comme un berceau d'affichage (et qui partage le système des utilisateurs et le système des permissions)
  - soit dépendant des autres modules (ex: Top10) et qui utilise le contenu des autres modules (du portail) et qui fait des opérations dessus
- Un module peut s'afficher:
  - soit dans un "block"
  - soit dans la place d'affichage centrale du portail
- Un module le plus souvent a deux interfaces:
  - interface administrateur pour gérer le module et qui s'intègre dans l'outil d'administration du système
  - interface utilisateur

2.1 "Tout est dans la doc ou presque :)"

Depuis le début vous avez besoin de:

- d'installer un portail test (fort risque de détruire votre portail en production)
- chercher et imprimer toute la doc (versions les plus récentes):

chercher ici:

url: http://www.postnuke.com/index.php
url: http://mods.postnuke.com/
url: http://centre.ics.uci.edu/~grape/

Voici la documentation la plus importante:

url: Postnuke Module Developers Guide
url: Postnuke API Command Reference
url: Postnuke ADODB Documentation

2.2 Etapes à suivre

- Imprimez la doc et lisez-la!!!
- S'inspirer des autres modules (proprement programmés)
- Décidez des fonctionalités de votre module et vérifiez qu'il n'existe pas déjà!
- Donnez un nom (court, tout en minuscules et qui fait du sens) et enregistrez-le dans le site officiel de postnuke pour éviter des conflits de nom avec d'autres modules
- Décidez du type de votre module: "item" ou "utility"
  - item = modules indépendants possédant leur propre contenu à manipuler (ex: news, FAQ, downloads)
  - utility= modules qui manipulent le contenu des autres modules (ex: comments, ratings) et qui utilisent souvent des "hooks" (interactions parmi modules)
- Faites le "design" de votre module
  - séparer bien les fonctions "user" et "admin"
  - séparer bien les fonctions "GUI" et "API" (affichage et opérations)
  - décidez si vous voulez avoir des "blocks"
- Réfléchissez bien pour la structure des tableaux dans la base de données si votre module en a besoin
- Programmez (selon la doc officielle et nos conseils :)
- Demandez de l'aide (*Module Support Forums*)
- Testez et debogguez, documentez (!), faites un "package" de votre module (zip, tar)

3. Programmation du module

Exemples pour s'inspirer:

url: Template **(module exemple de postnuke qui vient avec l'installation)**
url: joinproject **(simple)**
url: vquiz **(plus compliqué)**

3.1 Conseils importants

- Donnez des noms qui commencent avec le nom du module (ex: vquiz_questions). Cela inclut les noms de variables, de tableaux SQL et de fonctions.
- Codez en utilisant des fonctions (pas de partie "main"!). L'appel à ces fonctions se fait depuis un long URL du style:

```
 index.php?module=joinproject&type=admin&func=main
Paramètres: module(=nom du module), type(='user' ou 'admin'),
            func(=nom du fonction)
```

3.2 Fichiers importants

- ***Installation et initialisation:*** pninit.php, pntables.php
- ***Interface admin:*** pnadmin.php, pnadminapi.php
- ***Interface utilisateur:*** pnuser.php, pnuserapi.php
- ***Gestion des langues:*** répertoire pnlang/(eng, fra, deu, ...)/admin.php,user.php

3.3 Structure d'un module postnuke

- Tous les modules se trouvent dans le répertoire "modules"
- Tous les fichiers d'un module se trouvent dans un sous-répertoire avec le même nom avec celui du module
- La structure du répertoire d'un module devait être comme ci-dessous:

```
modules/
 vquiz/
     pninit.php
     pntables.php
     pnadmin.php
     pnadminapi.php
     pnuser.php
     pnuserapi.php
     pnblocks/
         snapshot.php
     pnimages/
         admin.png
     pnlang/
         eng/
             admin.php
             user.php
             init.php
             manual.html
             snapshot.php
     pnversion.php
```

Détails:

```
modules/
 vquiz/  ------------------------------> répertoire du module
     pninit.php      init
     pntables.php  ----------------------> initialisation et installation des tables
     pnadmin.php     admin
     pnadminapi.php ---------------------> fonctions GUI et API pour l'administrateur
     pnuser.php      user
     pnuserapi.php ----------------------> fonctions GUI et API pour l'utilisateur
     pnblocks/ --------------------------> fonctions pour créer un "block"
         snapshot.php
     pnimages/ --------------------------> images du module (ex: icon pour l'admin)
         admin.png
     pnlang/ ----------------------------> support multi-langues du module
         eng/
             admin.php     langues
             user.php  ------------------> définition des constants pour le texte
             init.php
             manual.html
             snapshot.php
     pnversion.php
```


4. Installation et initialisation

Postnuke a un interface pour initialiser, activer/désactiver ou enlever un module et il y a deux fichiers qui contribuent: "pntables.php" et "pninit.php".

4.1 Décrire toutes les tables SQL dans le fichier "pntables.php"

Il s'agit juste d'une fonction qui remplit l'array "pntable" avec le nom de toutes les tables utilisées et de leurs colonnes (ex ci-dessous: table "pn_joinproject_members" avec 4 colonnes)

```
function joinproject_pntables() {
    // Initialise table array
    $pntable = array();
    // Get the name for the template item table.
    $members = pnConfigGetVar('prefix') . '_joinproject_members';
    // Set the table name
    $pntable['joinproject_members'] = $members;
    // Set the column names.
    $pntable['joinproject_members_column'] =
array('memberid'   => $members . '.memberid',
      'projid'     => $members . '.projid',
      'membername' => $members . '.membername',
      'accepted'   => $members . '.accepted');
    // Return the table information
    return $pntable; }
```


4.2 Créer ou effacer les tableaux SQL dans le fichier "pninit.php" (fonctions "init()" et "delete()")

Dans init() on crée les tables dans la base de données avec des requêtes SQL

```
function joinproject_init() {
    list($dbconn) = pnDBGetConn();
    $pntable = pnDBGetTables();
    $memberstable = $pntable['joinproject_members'];
    $memberscolumn = &$pntable['joinproject_members_column'];
    $sql1 = "CREATE TABLE $memberstable (
            $memberscolumn[memberid] tinyint unsigned NOT NULL auto_increment,
            $memberscolumn[projid] tinyint unsigned NOT NULL default '',
            $memberscolumn[membername] varchar(255) NOT NULL default '',
            $memberscolumn[accepted] tinyint(1) NOT NULL default '0',
            PRIMARY KEY(memberid))";
   $dbconn->Execute($sql1);
    // Check for an error with the database code, and if so set an
    // appropriate error message and return
    if ($dbconn->ErrorNo() != 0) {
        pnSessionSetVar('errormsg', _CREATETABLE1FAILED);
        return false;
    }
}
```


Dans delete() on efface les tables de la base de données avec la requête SQL "DROP TABLE"

```
function joinproject_delete()
{
    list($dbconn) = pnDBGetConn();
    $pntable = pnDBGetTables();
    $sql1 = "DROP TABLE $pntable[joinproject_members]";
    $dbconn->Execute($sql1);
    // Check for an error with the database code, and if so set an
    // appropriate error message and return
    if ($dbconn->ErrorNo() != 0) {
        pnSessionSetVar('errormsg', _DROPTABLE2FAILED);
        return false;
    }
    // Deletion successful
    return true;
}
```

4.3 Tester (et vérifier la tables SQL dans la DB !)

Administration -> Modules -> Liste -> "MyModule" ->

Initialiser,

Activer/Désactiver,

Enlever

5. Interface admin et utilisateur

Interface admin:

- accès à l'administration du module depuis l'administration du portail

Interface utilisateur:

- soit dans un menu (ex: "Main Menu") qu'on appelle avec: {mymodule}
- soit dans un "block"
- et plusieurs d'autres possibilités

5.1 Séparer fonctions GUI et API

GUI: pnadmin.php, pnuser.php

API: pnadminapi.php, pnuserapi.php

5.2 tout en fonctions ou classes

Dans les fichiers *pnadmin.php* et *pnadminapi.php*:

- mymodule_admin_main(), mymodule_admin_function()

Dans les fichiers *pnuser.php* et *pnuserapi.php*:

- mymodule_user_main(), mymodule_user_function()

5.3 Utiliser les fonctions API (doc!)

pour appeler une fonction: pnModURL()

```
pnModURL('joinproject', 'user', 'main')
```

pour rediriger la page: pnRedirect()

```
pnRedirect(pnModURL('joinproject', 'admin', 'main'));
```

5.4 Pour SQL utiliser la librairie ADODB (doc!)

Postnuke utilise pour SQL la librairie ADODB et ci-dessous on voit le minimum des commandes dont on a besoin:

```
list($dbconn) = pnDBGetConn();
$pntable = pnDBGetTables();
$projectstable = $pntable['joinproject_projects'];
$result1=$dbconn->Execute("SELECT * FROM $projectstable");
```

5.5 Gérer les exceptions (doc!)

Postnuke a un système de gestion des exceptions: "system" et "user" et il est recommandé de l'utiliser.

5.6 Sécurité et permissions

Postnuke a un système assez évolué pour gérer les permissions (voir le manuel en ligne). On les gère facilement grâce à une fonction de l'API et on suggère de le faire au début de chaque fonction pour éviter les trous de sécurité:

```
if (!pnSecAuthAction(0, 'joinproject::', '::', ACCESS_ADMIN)) {
  $output->Text(_TEMPLATENOAUTH);
  return $output->GetOutput(); }
```

Il y a plusieurs niveaux d'accès:

```
ACCESS_NONE No access
ACCESS_OVERVIEW Allowed to get an overview of the content
ACCESS_READ Allowed to read the content
ACCESS_COMMENT Allowed to comment on the content
ACCESS_MODERATE Allowed to moderate the content
ACCESS_EDIT Allowed to edit the content
ACCESS_ADD Allowed to add content
ACCESS_DELETE Allowed to delete content
ACCESS_ADMIN Full access
```


5.7 Pour HTML utiliser l'objet pnHTML et ses méthodes (doc!)

Jamais de "echo" ou "print", à la place utilisez les méthodes du pnHTML.

Il y en a pour tout les éléments HTML.

```
$output = new pnHTML();
$output->Start();
$output->End();
$output->TableStart();
$output->TableEnd();
$output->TableAddRow();
$output->Text();
$output->Title();
$output->BoldText();
$output->FormStart();
$output->FormEnd();
$output->FormText();
$output->FormTextArea();
$output->FormHidden();
$output->FormList();
$output->FormSubmit();
$output->PrintPage();
$output->setInputMode();
$output->setOutputMode();
$output->Redirect();
$output->LineBreak();
$output->URL();
return $output->GetOutput();
```

6. Module multi-langues

Pour faire votre module multi-langues il faut faire:

- remplacer dans votre code tout le texte avec des constantes (ex: _MESSAGE)
- créer dans le répertoire "pnlang" du module un sous-répertoire pour la langue que vous voulez (ex: "fra") et à l'intérieur les fichiers "admin.php" et "user.php".

Exemple 6-1: Fichier mymodule/pnlang/fra/admin.php

```
<?php
define("_MODULETITLE","Join Project module");
define("_ADMININTERFACE","Interface administrateur");
define("_LISTOFPROJECTS","Liste des projets");
?>
```

Exemple 6-2: Fichier mymodule/pnlang/fra/user.php

```
<?php
define("_MODULETITLE","Join Project");
define("_JOINAPROJECT","Joindre un projet");
define("_LISTOFPROJECTS","Liste des projets");
?>
```

Exemple 6-3: Utilisation de ces constants dans pnadmin.php ou pnuser.php

```
$output->Title(_MODULETITLE);
```