Développer un module phpWebSite

Code: portal-prog-pws

Originaux

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

Auteurs et version

• Stéphane Morand - Daniel K. Schneider - Vivian Synteta
• Version: 0.1 (modifié le 24/6/02 par SM)

Prérequis

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

Abstract

Ce document décrit les bases du fonctionnement interne de phpWebSite et donne quelques conseils pour développer un module soi-même.

Objectifs

• Décrire l'architecture et le fonctionnement interne d'un module phpWebSite
• Etablir une liste d'étapes à suivre pour en developper un soi-même
• Donner quelques conseils

1. Table des matières détaillée

2. Introduction

2.1 Définitions

phpWebSite est un portail communautaire écrit en php disponible sur le site

url: http://phpwebsite.appstate.edu

Son architecture permet de personaliser le portail en ajoutant des modules qui augmentent les fonctionalités du portail. Un certain nombre de ces modules sont disponibles sur Internet, mais il est également possible de les créer soi-même.

2.2 Conseils de base

Le conseil le plus important est de bien lire la documentation: celle fournie avec la distribution du portail et celle disponible sur la homepage de phpWebSite.

Une bonne démarche est aussi de s'inspirer des modules les plus "officiels" qui sont fournis avec la distribution du portail.

Prendre le temps de refléchir aux fonctionalités que vous souhaitez avoir, regardez ce qui existe déja avant de vous lancer tête baissée.

3. Comment ca marche?

Il est important de comprendre la logique de base du fonctionnement interne des modules et de phpWebSite.

3.1 La structure d'un module

- Tous les modules se trouvent dans le répertoire module
- Chaque module se trouve dans un sous répertoire portant le nom du module
- Par exemple pour le module *friends*

```
mod/
    friends/
            index.php
            friends.php
            friends_config.php
            friends_block.php
            friends_functions.php
            friends_install.php
            friends_uninstall.php
            friends.gif
```


3.2 index.php

Le fichier index.php d'un module est son moteur. C'est ici que les actions concernant le modules sont faites.

```
if ($admintest==$security_hash)
// On est dans le traitement des fonctions administrateur
switch($op)
{
    case "install" :
         include("./mod/friends/friends_install.php");
         break;
    case "uninstall"" :
         include("./mod/friends/friends_uninstall.php");
         break;
    ...
}
// on est dans le traitement des actions de l'utilisateur
switch($op){
    case "fonction_utilisateur":
         action_utilisateur();
    ...
    default: afficher_page_acceuil_module()
}
```


3.3 installation/désinstallation

Les fonctions `friends_install.php` et `friends_uninstall.php` contiennent le code pour créer les bases de données, les peupler (ou les effacer) et ajouter un lien dans les tables de phpWebSite **`module`** et **`menu`** pour être atteignable.

Penser à préfixer les tables avec le préfixe de phpWebSite suivit de *`mod_<le nom de votre module>`*

3.4 les autres fichiers du modules

Les autres fichiers:

- *`friends.php`* contient les classes et fonctions principales du module.
- *`friends_config.php`* contient les variables de configuration du module
- *`friends_block.php`* est utilisé pour afficher le module à plus d'un endroit.

3.5 mod.php

A la racine de votre installation de phpWebSite se trouve le fichier **`mod.php`**.

il centralise toutes les commandes et éxecute le code du module en cours d'utilisation et si aucun c'est utilisé il redirige sur la page d'acceuil.

```
<?php
foreach ($HTTP_GET_VARS as $key=>$value){
  if (preg_match("/^\<script/", $value)){
    $HTTP_GET_VARS[$key] = NULL;
        $$key = NULL;
  }
}

global $current_mod, $current_op;
if($mod)
{
        $current_mod = $mod;
        $current_op = $op;
        include("./mod/$mod/index.php");
}
else Header("Location: ./index.php");
?>
```

4. Ecrire un module

4.1 Les conventions d'écriture de code

- Utiliser des préfixes pour les variables: **`OBJ_`** (objets), **`CNT_`** (contenu d'un module) **`SES_`** (variables de sessions), **`CLS_`** (classes). Par exemple: **`OBJ_friends = new CLS_friends`**
- Garder un bout du nom du module dans chacun des éléments qui s'y rapportent.
- Préfixer les tables de la base de données par **`mod_<le nom du module>`**
- Utiliser **`<?php`** (et non pas **`<?`** ou **`<?PHP`**)
- Séparer la logique de la présentation
- Utiliser le caractère '_' pour séparer les noms
- Eviter d'inclure des fichiers inutiles
- ... Voir la documentation de phpWebSite

4.2 La documentation

Bien documenter son code est important pour la lisibilité de son programme pour soi-même et pour les autres. L'utilisation de l'outil PHPDoc

url: http://www.phpdoc.de

Cet outils permet de générer (si le code à été commenté correctement) de la documentation au format HTML.

Voir la documentation de phpWebSite et de PHPDoc

4.3 Les fonctions existantes

Plusieurs fonctions sont utilisables dans phpWebSite. Elles peuvent gérer:

- Les permissions: **`$OBJ_user->allow_access()`**
- Le support des langues : **`$translate->it("texte à traduire")`**
- Le layout : **`thememainbox($title, $content)`**
- Les formulaires HTML, ...

5. Conclusion

La programation de module dans phpWebSite est relativement simple. L'utilisation est souple et même s'il est conseilllé de suivre les recommandations pour avoir plus de chance de fonctionner dans des versions futures, le développeur à pas mal de libertés. Le meilleur conseil à suivre est de faire preuve de bon sens.

Lire la dernière version de la documentation est très important car cela évolue très vite.