Introduction à la notion de module

01 : Pourquoi créer un module

Drupal est open-source alors pourquoi ne pas le modifier directement ?

Drupal propose un framework qui permet de disposer d'un nombre important de fonctionnalités qui ont été optimisées. Introduire des modifications pourrait nuire à ce fonctionnement. On se priverait des améliorations apportées par les nouvelles versions, de plus, ce code serait difficilement réutilisable.

Un module permet d'avoir tout le code spécifique à un projet dans un seul endroit et d'utiliser la puissance des API ce qui permet l'évolutivité, la portabilité et la fiabilité

Les concepts concernant les modules sont faciles à comprendre et à utiliser grâce aux 250 fonctions-hook et aux nombreux API.

02 Configuration du répertoire du module.

Le répertoire du module est un sous-répertoire du répertoire sites/all/modules (ou sites/default/modules ou encore sites/monsite.com/modules), ayant le même nom que le module.

Sa structure est la suivante :

- Obligatoirement 2 fichiers d'extension .info et .module qui ont le même nom que le module lui même.
- Un fichier optionnel README.txt.

Le module apparaît alors dans la liste des modules dans la page de gestion des modules.

03 Le fichier .info

Exemple de fichier : module1.info

```
name = Module numéro 1
description = "Premier exemple de module"
package = Cours Drupal DF
core = 7.x
files[] = module1.module
```

Contenu du fichier .info :

Ensemble de paires clé-valeur rendant accessible à Drupal un certain nombre de méta-données concernant notre module.

Même syntaxe que certains fichiers de configuration (Apache, PHP)

Explications :

name et description : utilisées dans la liste des modules
package = groupe de modules auquel il appartient
core =version de Drupal
files : tableau de fichier qui permet d'inclure des fichier dans le code_registry
Ceux-ci seront chargés conditionnellement dans le code en fonction des besoins

Dans la liste des modules (admin/modules/list) nous avons un groupe dont le nom correspond au nom du package dans lequel apparaît notre nom de module sa version et sa description.

04 Le fichier .module

Exemple de fichier : module1.module

```
<?php
/**
 * Implements hook_menu().
 */
function module1_menu() {
  $items['module1-page1'] = array(
    'page callback' => 'module1_page1',
    'access arguments' => array('access content'),
    'type' => MENU_CALLBACK,
  );
  return $items;
}

function module1_page1() {
  $page_array = array();  //page sous forme d'array
  $page_array[] = array(  //élément de page sous forme de render array
    '#type' => 'markup',
    '#markup' => 'Bienvenue sur ma première page Drupal',
  );
  return $page_array;
}
```

Remarque : On aurait simplement pu renvoyer une simple chaîne de caractère en faisant :

```
  return 'Bienvenue sur ma première page Drupal';
```

Ce fichier contient 2 fonctions.
Les fonctions utilisées dans Drupal respectent des conventions.

- Leur nom est préfixé avec le nom du module suivi du caractère '_'.
  C'est une forme de namespace pour éviter d'écraser d'autres fonctions.

- Certaines fonctions sont des hook dont le nom contient un suffix qui est un mot réservé (tel que menu)
  Ces fonctions-hook sont exécutées à certains moments spécifiques dans tous les modules actifs pour altérer ou étendre le fonctionnement de Drupal.

Le hook-menu permet d'ajouter des pages à un site Drupal en déclarant les URL qui permettront d'accéder à ces pages mais aussi les fonctions qui vont construire ces pages.
Les hook sont mis en évidence dans le code par un commentaire : Implements hook_HOOK

Explication de l'exemple :
Ici nous voulons créer une page simple en donnant son chemin et la fonction qui construit son contenu.

Nous définissons un **tableau $items** dont les valeurs sont des tableaux imbriqués.

La **clé** correspond au **chemin** dans l'URL qu'il faut visiter pour voir notre page.

Dans le tableau imbriqué nous avons un ensemble de paires clé-valeur qui constituent la configuration de cette page.
La première est '**page callback**' qui donne le **nom de la fonction** qui sera appelée pour construire le contenu de la page.
La seconde est '**access arguments**' et permet de définir la permission qui est nécessaire pour voir cette page.
Le '**type**' est MENU_CALLBACK. Il indique que notre page existe mais n'est pas référencée dans un menu.
D'autres types existent qui créent la page mais **ajoutent également des liens** vers notre page dans certains **menus prédéfinis**.
Finalement nous renvoyons le tableau $items.

Attention ! Le terme « Menu » peut prêter à confusion. Ici, menu ne signifie pas « élément d'interface formé de liens hiérarchisés et permettant d'accéder aux pages d'un site Web ». Menu est à prendre dans le sens « menu d'un restaurant » c'est à dire la carte de tous les plats pouvant être commandés par un client. Le menu est donc la liste de toutes les URLs proposées par le site.

Drupal utilise un **menu_registry** (objet en cache dans la BD) qui permet de ne pas exécuter ces fonctions **menu** à chaque appel de page. Il contient les items de menu de tous les modules actifs.
Quand un hook_menu à été modifié, il faut donc recréer le cache. Tous les hook-menu de tous les modules sont alors ré-exécutés pour que les nouvelles pages soient ajoutées au menu_registry.
Pendant l'activation d'un module le menu_registry est automatiquement reconstruit pour ajouter les URLs définies par ce module.

La seconde fonction renvoie le contenu de la page. Toute page d'un site Drupal correspond à une fonction.

Exercice :
1) Dans votre site Drupal 7, créez le module **module1** avec :
- son répertoire
- son fichier .info
- son fichier .module

2) Activez module1.

3) Dans votre browser completez l'adresse de base de votre site Drupal avec /module1-page1 pour faire afficher la page que nous avons définie.

Problème : le module ne s'active pas correctement ou une page ne s'affiche pas.
Cause 1 : Certains fichiers ne sont pas encodés en UTF8.
Solution : ouvrir chaque fichier du module avec notepad++ et vérifier qu'en bas à droite il y a bien écrit ANSI as UTF-8. Si ce n'est pas le cas, il faut convertir le fichier : Menu Encodage / Option Convertir en UTF-8 (sans BOM). Il faut bien utiliser Convertir et pas Encoder qui ne fait que modifier l'affichage et pas le fichier. Ensuite on sauvegarde le fichier. Maintenant que les fichiers sont corrects, il faut : Désactiver, puis désinstaller, puis réactiver le module.
Cause 2 : Le hook menu a été modifié après l'activation du module. Solution : faire un "Clear All Caches".
Cause 3 : Les "Clean Urls" ne sont pas activées. Solution : ajouter ?q= avant le chemin de la page.