
PHP & Bases de données non relationnelles

Mohammad Ghoniem
Université de Bretagne Sud

Plan

- I. Programmation orientée objet
- II. La manipulation de chaînes de caractères
- III. La manipulation de fichiers
- IV. Les bases de données non relationnelles

I. Programmation orientée objet en PHP

-
1. Définition d'une classe
 2. Instanciation et héritage

Création d'une classe

- Instruction *class*
- Définition des variables d'instance
- Définition des méthodes d'instance
- Différence avec les langages orientés objet classiques
 - Limitation à un seul constructeur
- Opérateur `->` pour accéder aux variables et aux méthodes d'un objet

Instanciación et héritage

- \$mavariabile = *new* MaClasse(\$arg);
- *class* File *extends* Mere

II. Manipulation de chaînes de caractères

-
1. Fonctions simples
 2. Expressions régulières

Manipulation de chaînes de caractères

- Utile dans de nombreuses applications
 - Validation de saisie
 - Extraction de données d'un fichier texte

Fonctions de chaînes simples

- `substr()`
 - renvoie un fragment de chaîne
- `trim()`
 - supprime les blancs au début et à la fin
- `chr()` et `ord()`
 - conversion code ascii ↔ caractère
- `strlen()`
 - renvoie la longueur de la chaîne
- `printf()` et `sprintf()`
 - mise en forme de la chaîne
- `number_format()`
 - mise en forme simple

Expressions régulières

- `^modele` identifie toute chaîne commençant par `modele`
- `modele$` identifie toute chaîne finissant par `modele`
- Séquences d'échappement : `\t` , `\n`, `\r`

Expressions régulières

■ Les classes de caractères

- ❑ [a-z] identifie toute minuscule
- ❑ [A-Z] identifie toute majuscule
- ❑ [a-zA-Z] identifie toute lettre
- ❑ [0-9] identifie tout chiffre
- ❑ [0-9\.\-] identifie tout chiffre, point, ou moins
- ❑ [\f\r\n\t] identifie tout caractère vide
- ❑ [[:alpha:]], [[:digit:]], [[:alnum:]], [[:space:]], [[:upper:]], [[:lower:]], [[:punct:]], [[:xdigit:]]

Exemples

- [AaEeliOoUu]
- ^[a-z][0-9]\$
- ^[^a-z][0-9]\$
- [^\\ \\ ^]
- [^\\ \" \']

Détecter des occurrences multiples

- $^{\wedge}[[[:\text{alpha}:]]\{3\}\$$
- $^{\wedge}\text{a}\$$
- $^{\wedge}\text{a}\{3\}\$$
- $^{\wedge}\text{a}\{2, 4\}\$$
- $^{\wedge}\text{a}\{2,\}\$$
- $^{\wedge}\text{a}\{2,\}$
- $\text{a}\{2,\}$
- $\{2\}$

Détecter des occurrences multiples

- $^{\wedge}[a-zA-Z0-9_]{1,}\$ \Leftrightarrow ^{\wedge}[a-zA-Z0-9_]+\$$
- $^{\wedge}[0-9]{1,}\$ \Leftrightarrow ^{\wedge}[0-9]+\$$
- $^{\wedge}\{-\{0,1\}[0-9]{1,}\}\$ \Leftrightarrow ^{\wedge}\{-?\{0-9\}+\}\$$
- $^{\wedge}\{-\{0,1\}[0-9]{0,}\}\.\{-\{0,1\}[0-9]{0,}\}\$$
 $\Leftrightarrow ^{\wedge}\{-?\{0-9\}*\}\.\{-?\{0-9\}*\}\$$
- $^{\wedge}\{.\{+\}\@.\{+\}\}\.\{+\}\$$

Les alternatives

- $[ac] \Leftrightarrow a|c$
- $(fak|saph)ir$
- $assis|clous\$$ différent de $(assis|clous)\$$
- $([wx])([yz])$

Fonctions PHP utilisant les expressions régulières

- **ereg() et eregi()**
 - int `ereg(string modele, string source, tableau [regs])`
 - `eregi()` ignore la casse
- **ereg_replace() et eregi_replace()**
 - string `ereg_replace(string modele, string rempl, string chaine)`
- **split()**
 - array `split(string delimitateur, string chaine, int [limit])`
- **sql_regcase()**
 - convertit une expression régulière sensible à la casse en une autre insensible à la casse

III. Manipulation de fichiers et stockage de données

Gestion de fichiers

■ Ouverture

- ❑ `int fopen(string fichier, string mode)`

- ❑ Modes possibles :

- `a` et `a+` (ajout , ajout + lecture)

- `r` et `r+` (lecture seule, lecture + écriture)

- `w` et `w+` (écriture seule, écriture + lecture)

- ❑ Exemple :

```
if(!$fichier=fopen("picture.gif", "rb")){  
    echo("Ouverture de fichier impossible");  
}
```

Gestion de fichiers

■ Fermeture

- ❑ `int fclose(int pf);`

■ Affichage

- ❑ `int fpassthru(int pf);`

■ Lecture

- ❑ `string fread(int pf, int longueur)`

// lit une chaîne

- ❑ `string fgetc(int pf);`

// lit un caractère

- ❑ `string fgets(int pf, int longueur)`

// lit une chaîne

- ❑ `string fgetss(int pf, int longueur)`

// ignore les balises

- ❑ `array file(string nom)`

// place le contenu

// du fichier dans un

// tableau

Gestion de fichiers

■ Écriture

- ❑ `int fputs(int pf, string chaine, int [longueur]);`
- ❑ `int fwrite(int pf, string chaine, int [longueur]);`

■ Navigation

- ❑ `int rewind(int pf);` // retour au début du fichier
- ❑ `int fseek(int pf, int offset);`// vers une position spécifique
- ❑ `int ftell(int pf);` // détermine la position courante
- ❑ `int feof(int pf);` // vrai si fin de fichier

■ Copie

- ❑ `int copy(string source, string destination);`

Gestion de fichiers

- **Suppression**
 - `int unlink(string nom);`
- **Renommage**
 - `int rename(string ancien, string nouveau);`
- **Identification des attributs de fichier**
 - `int file_exists(string nom);`
 - `int file_size(string nom);`
 - `string file_type(string nom);`
 - `is_dir(), is_executable(), is_file(), is_link(), is_readable(), is_writable()`

Gestion de répertoires

- `int chdir(string nom);` // définit le répertoire
// courant
- `int opendir(string chemin);` // ouverture
- `string readdir(int pointeur);` // lecture
- `void rewinddir(int pointeur);` // retour au début
- `void closedir(int pointeur);` // fermeture
- Syntaxe objet possible
 - `$rep = dir("/temp");`
 - `$rep->read(), $rep->rewind() et $rep->close()`
- `int mkdir(string chemin, int mode);` // création
- `int rmdir(string nom);` // suppression

Chargement et gestion de fichiers

■ Chargement

```
<FORM ACTION="upload.php" METHOD="POST"  
  ENCTYPE="multipart/form-data">  
<INPUT TYPE="FILE" NAME="userfile"><BR>  
<INPUT TYPE="SUBMIT"><BR>  
</FORM>
```

■ Gestion des fichiers chargés

- enregistrés sous le nom phpX dans le répertoire temporaire
- Détruits automatiquement à la fin de la requête
- Doivent être copiés pour un usage ultérieur

IV. Les bases de données non relationnelles

Étude de cas
Création d'un carnet d'adresses

Les bases de données

- Deux catégories de BDD
 - Relationnelles
 - à base de tables reliées entre elles par des clés
 - Non relationnelles
 - ignorent la structure
 - souvent implémentées dans des fichiers
- Les BDD non relationnelles
 - prises en charge par l'interface DBA de PHP
 - stockent les données sous formes de paires clé/valeur sous forme de chaînes de caractères
 - ne nécessitent aucun logiciel supplémentaire
- L'interface DBA supporte toutes les opérations d'accès aux BDD

Création d'un carnet d'adresses

■ Objectifs

- ❑ Éditer, supprimer et afficher des enregistrements existants,
- ❑ Créer de nouveaux enregistrements,
- ❑ Effectuer des recherches dans la base de données,
- ❑ Interface modulable pour répondre à de nouveaux besoins,
- ❑ Fonction d'importation à partir de fichiers CSV : nécessité de sérialiser et désérialiser les enregistrements

Interface utilisateur

- Un formulaire HTML permettant l'accès aux fonctionnalités du carnet d'adresse
- Affichage à l'aide de la fonction `afficher_menu()`

Préserver un aperçu des données

- Fonctions `action_vue()`, `vue_start()`, `vue_entry()` et `vue_end()`
- Présentation sous forme de table HTML :
une ligne = un enregistrement

Ouvrir la base de données

- `int dba_open(string path, string mode, string gestionnaire, int [mode]);`
 - `path` : chemin vers le fichier
 - `mode` : `r`, `w`, `c`, `n` (lecture, écriture, création, écrasement)
 - `gestionnaire` : `dbm`, `ndbm`, `gdbm`, `db2`, `cdb`
 - 4ème paramètre optionnel : mode de la bdd créée
- Ces quatre paramètres seront stockées dans des variables pour un usage ultérieur

Parcourir la base de données

- initialiser la vue à l'aide de `vue_start()`
- `dba_firstkey(int pointeur);` // récupère la première clé de la BDD
- `dba_nextkey();` // itère sur la BDD
- `string dba_fetch(string clé, int pointeur);` // récupère la valeur associée à la clé
- désérialiser la valeur à l'aide de `unserialize()` et la passer à `vue_entry()`
- terminer avec `vue_end()`

Chercher dans la base de données

■ Fonctions nécessaires

- ❑ fonction `recherche_motcle($db, $motcle)`
qui effectue une recherche dans toute la base
puis affiche les enregistrements correspondants
- ❑ Fonction `action_recherche($motcle)`
qui ouvre la base de données et appelle la
fonction précédente, puis réalise l'affichage des
résultats

Supprimer un enregistrement

- fonction `action_supprime($id)`
 - ❑ Ouverture de la BDD
 - ❑ Suppression à l'aide de `dba_delete($id, $db)`
 - ❑ Entérinement par `dba_sync($db)`
 - ❑ Fermeture de la BDD

Afficher un enregistrement

- fonction `affiche_entree($data)`
 - affichage HTML
- fonction `action_voir($id)`
 - extraction dans la BDD

Éditer le carnet d'adresses

- fonction `edit_form($donnee)`
 - formulaire de saisie HTML
- fonction `action_edit($id)`
 - extrait l'enregistrement à partir de la BDD
 - transmet les données à `edit_form()`

Mettre à jour la BDD

- fonction `action_maj($donnee)`
 - ❑ Ouverture de la BDD
 - ❑ Remplacement de l'enregistrement à l'aide de `dba_replace(string clé, string valeur, int pointeur)`
 - ❑ Synchronisation de la BDD
 - ❑ Fermeture de la BDD

Ajouter une entrée

- fonction `prochain_id($db)`
 - extraction du prochain identifiant disponible
- fonction `action_nouveau()`
 - ouverture de la BDD, si elle existe
 - création de la BDD, sinon
 - extraction du prochain identifiant disponible
 - utilisation du formulaire de mise à jour pour l'ajout

Importer un fichier CSV

- fonction `csv_import($path)`
 - parcourt entièrement le fichier
 - l'analyse ligne par ligne à l'aide de `fgetcsv()`
 - ajoute les données dans la BDD
 - retourne le nombre d'enregistrements importés
- fonction `action_csv_import()`
 - stocke le fichier importé depuis le client dans `$csvfile`
 - appelle la fonction d'import

Implémentation de la partie principale

- Configuration du chemin d'accès à la base et du mode d'accès
- Déclaration des variables globales
- Affichage du menu initial
- Récupération de l'action à exécuter
- Aiguillage sur la méthode voulue (édition, suppression etc.)