Technologie Web
Chapitre 3 : PHP

Cecilia Zanni-Merk

cecilia.zanni-merk@insa-rouen.fr

Bureau BO B R1 04

Contenu

- Introduction à PHP
- Syntaxe
- Fonctions et modularité
- Chaînes de caractères
- Les formulaires
- Les fichiers
- Les classes
- Les exceptions
- Les sessions
- Bases de données

Références

- Cours « Technologies Web » d'Alexandre Pauchet, INSA Rouen Normandie, 2016
- MOOC « Soyez Acteurs du Web », https://www.fun-mooc.fr/
- Olivier HEURTEL. PHP Développer un site Web dynamique et interactif. ENI éditions. ISBN 978-2-7460-3992-6

Les fichiers

Gestion de fichiers

- **Ouverture :** `$fichier=fopen("NOM", "MODE");` avec MODE valant :
  - `r, r+ :` lecture et lecture/écriture, pointeur au début
  - `w, w+ :` écriture et lecture/écriture, avec création ou effacement, pointeur au début
  - `a, a+ :` écriture et lecture/écriture, pointeur à la fin, avec création
  - `x, x+ :` création en écriture et lecture/écriture, pointeur au début, erreur en cas d'existence du fichier
  - `c, c+ :` création en écriture et lecture/écriture, pointeur au début, sans erreur
- **Verrouillage :** `bool flock($fichier, int $operation)`
- **Fermeture :** `fclose($fichier);`

Utilisation de fichiers

- Lecture d'une ligne :
  `string fgets($fichier [, integer nbOctets])`
- Lecture d'un caractère :
  `string fgetc($fichier)`
- Ecriture d'une ligne :
  `integer fputs($fichier, string)`
- Test de fin de fichier :
  `boolean feof($fichier)`
- Positionnement :
  `fseek($fichier, int $position);`

Exemple

- Téléchargez et testez le script `fichier.php`

Les classes

Déclaration

```
class NomClasse {
      public / protected / private $attribut1 [ = constante1 ];
      public / protected / private $attribut2 [ = constante2 ];
      ...
      public / protected / private function __construct (...) {
            // constructeur }
      public / protected / private function methode1 (...) {
            // méthode }
      public / protected / private function methode2 (...) {
            // méthode  }
      ...
}
```

- Seules les initialisations par constante sont autorisées

Attributs et méthodes

- Instanciation : `$objet = new NomClasse(...)`
- Accès aux attributs et méthodes par l'opérateur "`->`"

```
$objet->attribut / $objet->methode()
```

- L'accès aux attributs dans les méthodes se fait par

```
$this->attribut
```

- La surcharge des méthodes dans une même classe n'est pas possible, mais la redéfinition dans une classe fille l'est

Attributs et méthodes

- Instanciation : `$objet = new NomClasse(...)`
- Accès aux attributs et méthodes par l'opérateur "`->`"
  `$objet->attribut / $objet->methode()`
- L'accès aux attributs dans les méthodes se fait par
  `$this->attribut`
- La surcharge des méthodes dans une même classe n'est pas possible, mais la redéfinition dans une classe fille l'est

- Téléchargez et testez le script `objet.php`

Constructeurs et destructeurs

- PHP5 permet les constructeurs unifiés et les destructeurs :

```
void __construct ([ arguments ]) {...}
void __destruct () {...}
```

- En cas d'héritage, appel explicite du constructeur/destructeur de la classe mère dans le corps du constructeur de la classe fille :

```
parent :: __construct ([ arguments ]);
parent :: __destruct ();
```

Héritage

```
class ClassDerivée extends ClasseHéritée {
    ...
}
```

- si une classe dérivée n'a pas de constructeur, celui de la classe mère est appelé
- la propagation d'appel des constructeurs n'est pas automatique
- l'opérateur de résolution portée est "`::`"
- `parent` est un mot clef permettant l'accès à la classe mère

Exemple

- Téléchargez et testez le script `heritage.php`
- N’oubliez pas les fichiers inclus
  - `Persone.class.php`
  - `Etudiant.class.php`

Classes abstraites

- PHP5 permet la création de classes abstraites, ne permettant pas l'instanciation d'objets mais servant de classe de base pour la création de classes dérivées.
- `abstract` sert à déclarer les méthodes et classes abstraites.

- Téléchargez et testez le script `compte.php`

Les interfaces

- déclarées par le mot clé `interface`
- ne contiennent aucune déclaration d'attribut
- ne contiennent aucune implémentation de méthode
- dont les déclarations de méthodes sont `public`
- implémentées par une classe par `implements` ; une classe peut implémenter plusieurs interfaces.

- Téléchargez et testez le script `interface.php`

Méthodes et classes finales

- il est possible d'empêcher toute redéfinition de méthode ou de classe, par le mot clé `final`.

```
class UneClasse {
   final function methode ()
        { ... }
}
```

empêche toute redéfinition de `methode()` dans les classes dérivées.

```
final class UneClasse {
   ...
}
```

empêche toute dérivation de la classe `uneClasse`.

Clonage d'objets

- Les opérateurs d'affectation = et de référence & permettent de copier un objet, mais les modifications sur la copie sont répercutées sur l'original.
- Pour éviter cela, il faut cloner les objets par :

```
$objetclone = clone $objet ;
```

- Pour modifier les propriétés de l'objet cloné, il faut définir la méthode prédéfinie `__clone()`.

- Téléchargez et testez le script `clonage.php` avec ses sources associées `Clonage.inc.php`, `Etudiant.class.php`

Attributs et méthodes statiques

- Déclaration par le mot clé `static`
- Accès par ‘`nomclasse::`’

Attributs et méthodes statiques

- Déclaration par le mot clé `static`
- Accès par ‘`nomclasse::`’

- Téléchargez et testez le script `statique.php`

Astuces diverses

- La méthode `_toString():string` : permet de formater l'affichage d'une instance de la classe.
- *Sérialisation d'objets* : Pour faciliter l'enregistrement d'objets dans des fichiers il est conseillé d'utiliser les méthodes `serialize` et `unserialize` pour en enregistrer une représentation linéaire.

Les exceptions

La classe Exception

- PHP5 introduit la classe prédéfinie `Exception`
- 2 attributs :
  - `message` : message d'erreur (`string`)
  - `code` : code d'erreur facultatif (`int`)
- Méthodes :
  - `getMessage()` : accesseur sur le message de l'objet
  - `getCode()` : accesseur sur le code d'erreur de l'objet
  - `getFile()` : retourne le fichier contenant l'erreur
  - `getLine()` : retourne la ligne renvoyant l'exception
  - `__toString()` : retourne une chaîne descriptive de l'exception

Code type. Récupération des exceptions

```
try {
      // Code à surveiller
      // Ce code peut renvoyer des erreurs non de votre fait
      if( test erreur ) {
            throw new Exception ();
      }
      else {
            // Résultat ;
      }
  }
catch ( Exception $except ) {
      // Gestion des erreurs
}
```

Exceptions. Personnalisation

- Héritage d'exception : le mécanisme de l'héritage permet d'étendre la classe `Exception`

```
class MonException extends Exception
{
     public function alerte ( $mess ) {
     echo "<script type ='text/javascript'>
     alert ('Erreur n ".$this->getCode(). "<br/>".
     $this->getMessage()."<br/>". $mess .
     "')</script>";
  }
}
```

Exemple

- Téléchargez et testez le script `exception.php`

Les sessions

Sessions et authentification

- La gestion de session permet de stocker des données entre les différentes pages visitées
  - `session_start()` : crée ou restaure une session
    - À mettre obligatoirement avant tout envoi d'en-tête
    - À mettre dans toute page participant à une session
  - la variable superglobale `$_SESSION` permet l'enregistrement de variables dans la session :
    - `$_SESSION["nomdelavariable "]=x` ajoute ou modifie une variable à/de la session en cours
    - `$_SESSION["nomdelavariable "]` accède à la valeur d'une variable de la session en cours
    - `unset($_SESSION["nomdelavariable "])` retire la variable
  - `session_destroy()` détruit la session en cours

Sessions et authentification

- La gestion de session permet de stocker des données entre les différentes pages visitées
  - `session_start()` : crée ou restaure une session
    - À mettre obligatoirement avant tout envoi d'en-tête
    - À mettre dans toute page participant à une session
  - la variable superglobale `$_SESSION` permet l'en dans la session :
    - `$_SESSION["nomdelavariable "]=x` ajoute ou session en cours
    - `$_SESSION["nomdelavariable "]` accède à la valeur d'une variable de la session en cours
    - `unset($_SESSION["nomdelavariable "])` retire la variable
  - `session_destroy()` détruit la session en cours

L'utilisation d'une session ne nécessite pas forcément d'authentification

Authentification

- En général, on utilise un formulaire pour saisir un pseudo et un mot de passe
- Une fois le login et mot de passe récupérés, il faut vérifier si l'utilisateur a le droit de se connecter.
- Les infos peuvent être stockées dans un fichier ou dans une base de données.

Exemple

- Téléchargez et testez les scripts
  - `authentification-form.php`
  - `authentification-page1.php`
  - `authentification-page2.php`
  - `login.inc.php`
  - `logout.php`

Bases de données

SQLite

- PHP supporte un grand nombre de bases de données
  - Oracle, Sybase, Ingres II, MySQL, PostgreSQL, **SQLite (inclus dans PHP5)**, . . .
- Caractéristiques de SQLite
  - SGBD embarqué dans la distribution de PHP5 →) pas de processus indépendant
  - léger et rapide (pas d'architecture client-serveur)
- SQLite implémente la norme SQL 92
  - Classes facilitant son interaction avec PHP
  - BD en un et un seul fichier
  - pas d'insertions concurrentes (base verrouillée)
  - accès concurrents en lecture seule

Ouverture et fermeture

- Accès à la base
  - Ouverture : `$db = new PDO("sqlite:path/filename");`
  - Fermeture : `unset($db);`

- La syntaxe est complexe … A travailler avec la documentation officielle

- Téléchargez et testez (attention à créer la structure de sous-dossiers nécessaire à la création de la BD `./data/`)
  - `SQLite-Creation.php`
  - `SQLite-Insertion.php`
  - `SQLite-Selection.php`

Références

Webographie

- http://www.php.net/
- Documentation officielle SQLite
  - http://fr.php.net/manual/fr/ref.sqlite.php
  - http://sqlite.org/
- Le tutoriel SQLite du livre d'Oliver HEURTEL sur Moodle