Cours PHP

Jean-Marc Fontaine

1 Qu'est-ce que PHP ?

1.1 Introduction à PHP

PHP a été créé en 1994 par Rasmus Lerdorf, un programmeur Danois, pour son usage personnel. A l'origine il s'agissait d'une bibliothèque Perl dont il se servait pour gérer son CV en ligne. Il y a rapidement ajouté de nombreuses fonctionnalités avant de la réécrire en C.

En 1995, il décide de la publier afin que d'autres profitent de ses travaux. PHP s'appelait alors PHP/FI (pour Personal Home Page Tools/Form Interpreter).

En 1997, deux étudiants Andi Gutmans et Zeev Suraski, réécrivent totalement le noyau de PHP. Cela deviendra le Zend Engine qui sera la base de la version 3 de PHP. C'est d'ailleurs à la sortie de cette version que PHP/FI est raccourci en PHP pour PHP: Hypertext Preprocessor.

PHP est un langage de script[1], c'est à dire qu'il est interprété et non compilé contrairement à des langages comme C++ par exemple. Il est principalement utilisé pour produire des pages web mais il est possible de l'utiliser dans d'autres environnement comme la ligne de commande. Nous verrons cet usage particulier un peu plus tard.

1.2 Architecture

1.2.1 Parcours d'une requête HTTP

TODO : Faire schéma du parcours d'une requête HTTP

1.2.2 Fonctionnement interne de PHP

TODO : Faire schéma du fonctionnement interne de PHP

1.3 L'écosystème PHP

La simplicité de prise en main de PHP l'a très vite rendu populaire et au fil des années sont apparus de nombreux outils et d'innombrables librairies. La plupart sont disponibles sous des licences libres[2] qui permettent de les utiliser dans vos projets.

Nous allons évoquer certains de ces outils et de ces librairies.

1.3.1 Librairies

1.3.1.1 PEAR

PEAR[3], acronyme pour PHP Extension and Application Repository est un ensemble de paquets. Chaque paquet a pour vocation de fournir des classes PHP permettant de simplifier la gestion d'un aspect d'une application comme la connexion à la base de données, la gestion des formulaires ou encore la gestion avancée des dates.

L'installation des paquets PEAR peut se faire manuellement ou à l'aide d'un installeur assez

1 http://fr.wikipedia.org/wiki/Langage_de_script
2 http://fr.wikipedia.org/wiki/Licences_Libres
3 http://pear.php.net/

sophistiqué qui est capable de télécharger et d'installer les paquets mais également de fonctionnalités plus avancées comme la gestion dépendances.

1.3.1.2 PECL

PECL[4], pour PHP Extension Community Library, est la librairies soeur de PEAR mais pour les extensions PHP écrites en C.

Comme nous l'avons vu précédemment, l'architecture de PHP est très modulaire et il est possible d'écrire ses propres extensions en langage C. Celles-ci doivent être compilées avant de pouvoir être utilisées.

La version de PHP distribuée est composée d'un noyau et d'extensions. Les extensions qui ne sont pas retenues pour faire partie de la distribution officielle sont disponibles via PECL.

La compilation et l'installation d'une extension PECL se fait soit manuellement soit à l'aide du mme installeur que celui de PEAR.

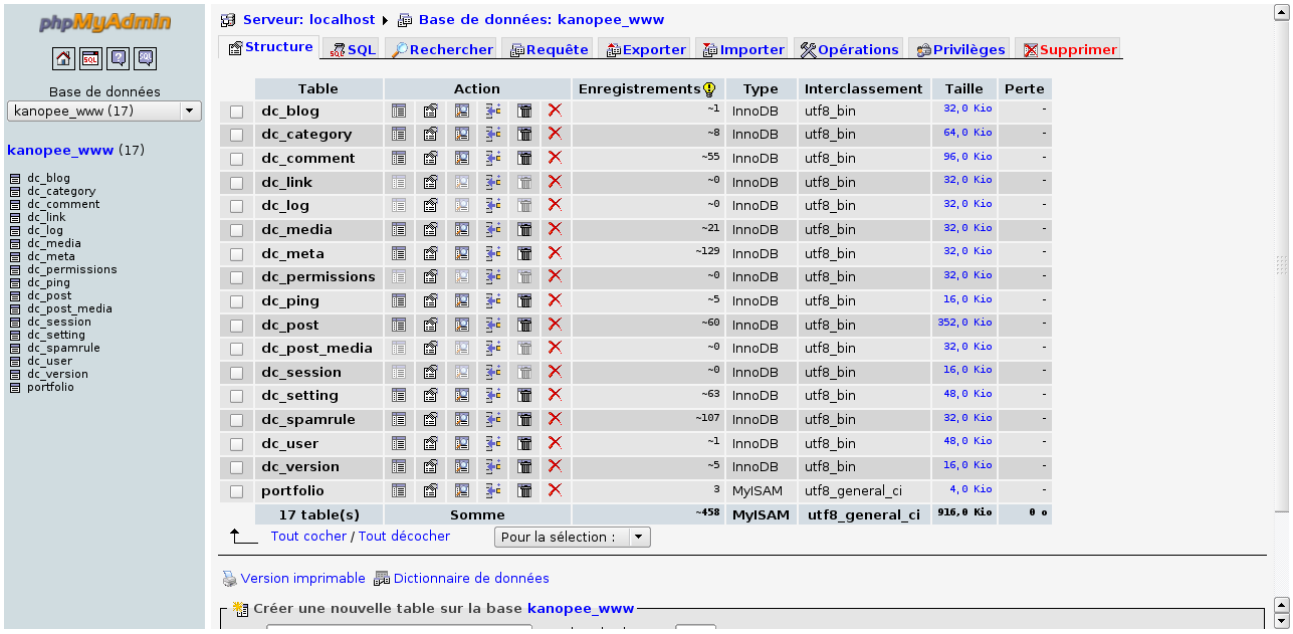
1.3.1.3 eZ Components

La librairie eZ Components[5], publiée par l'éditeur du CMS eZ Publish, n'est pas un framework à proprement parlé mais un ensemble de composants faiblement couplés qui permettent de simplifier la gestion de certains aspects d'une application comme la gestion du cache, la génération de graphiques ou encore la gestion de gabarits.

1.3.2 Outils

1.3.2.1 phpMyAdmin

phpMyAdmin[6] est une application PHP qui permet de gérer des bases de données MySQL. Cet outil est très populaire et est généralement disponible sur chez les hébergeurs.



4 http://pecl.php.net/
5 http://www.ez.no/fr/ezcomponents
6 http://www.phpmyadmin.net/

1.3.3 Frameworks

Les frameworks, ou cadres de travail en français, sont des outils qui permettent de simplifier les taches de bas niveaux afin que les développeurs se consacrent aux aspects métier de l'application qu'ils développent.

La plupart des langage en possèdent. On peut citer Struts pour Java, Ruby On Rails pour Ruby ou encore Django pour Python.

Il existe de nombreux frameworks PHP[7]. Nous allons évoquer rapidement les principaux.

1.3.3.1 CakePHP

CakePHP[8] est un framework inspiré de Ruby On Rails. Il utilise le Design Pattern ActiveRecord[9] pour accéder aux bases de données. Il utilise également un autre Design Pattern, Modèle-Vue-Contrôleur ou MVC[10], pour structurer les applications.

Voici ses principales caractéristiques :

- Compatible avec PHP4 et PHP5.
- Integration de CRUD pour l'utilisation simplifiée des bases de données SQL.
- Dispatcheur d'URL permettant d'obtenir des adresses aisément lisibles.
- Rapide et flexible avec un moteur de templates utilisant la syntaxe PHP et apportant des helpeurs.
- Helpeurs de vue permettant l'utilisation de Ajax, JavaScript, HTML, des formulaires et bien d'autres.
- Fonctionne dans n'importe quel sous-répertoire pour peu qu'il y soit accessible via un serveur HTTP tel que Apache.
- Contrôle des données.
- Composants de sécurité et de gestion des sessions.
- Cache des vues flexible.

1.3.3.2 Symfony

Symfony[11] est un framework PHP récent développé par une société française, Sensio. Comme CakePHP, il est basé sur la Design Pattern MVC.

Très en vogue depuis quelques temps, il a notamment été adopté par Yahoo! pour certains projets.

Voici ses principales caractéristiques :

- Une séparation du code en trois couches, selon le modèle MVC, pour une plus grande maintenabilité et évolutivité
- Un templating simple, basé sur PHP et des jeux de "helpers", ou fonctions additionnelles pour les gabarits...
- Des performances optimisées et un système de cache pour garantir des temps de réponse

7 http://fr.wikipedia.org/wiki/Liste_de_frameworks_PHP
8 http://www.cakephp.org/
9 http://fr.wikipedia.org/wiki/Active_record_%28patron_de_conception%29
10 http://fr.wikipedia.org/wiki/Mod%C3%A8le-Vue-Contr%C3%B4leur
11 http://www.symfony-project.org/

optimums

- Une gestion des url parlantes, qui permet de formater l'url d'une page indépendamment de sa position dans l'arborescence fonctionnelle
- Un système de configuration en cascade qui utilise de façon extensive le langage YAML
- Un générateur de back-office et un "démarreur de module" (scaffolding)
- Un support de l'I18N - symfony est nativement multi-langue,
- Une couche de mapping objet-relationnel (ORM) et une couche d'abstraction de données
- Le support de l'Ajax
- Une architecture extensible, permettant la création et l'utilisation de plugins

1.3.3.3 Zend Framework

Développé depuis 2006 par de nombreux contributeurs, le Zend Framework[12] a été lancé, comme son nom l'indique, à l'initiative de Zend, la société de Zeev Suraski et Andi Gutmans.

Il s'agit d'un framework souple dont le but est d'offrir une aide au développement de projets complexes sans pour autant brider les développeurs. C'est en effet, un reproche récurrents fait aux frameworks.

Pour cela, ses composants sont le moins couplés possibles. C'est à dire qu'il est possible de n'utiliser que ceux qui sont nécessaires à un projet ou de substituer un composant maison à un composant du framework afin de mieux coller aux besoins spécifiques du projet.

Voici ses principales caractéristiques :

- Code intégralement en PHP5 orienté objet et conforme au niveau d'erreur E_STRICT.
- Architecture souple avec des composants faiblement couplés et un minimum d'interdépendances.
- Implémentation MVC extensible supportant les habillages et les gabarits.
- Implémentation fexible du Design Pattern Table Gateway pour accéder aux données stockées dans des bases de données relationnelles depuis un environnement orienté objet.
- Support de nombreux systèmes de base de données dont MySQL, Oracle, IBM DB2, Microsoft SQL Server, PostgreSQL, SQLite et Informix Dynamic Server.
- Authentification et gestion des droits à l'aide des ACL.
- Filtrage et validation des données pour une meilleure sécurité des applications.
- Gestion des sessions.
- Composant de gestion de la configuration pour assurer une homogénéité au sein de applications.
- Création et envoi de mails.
- Indexation et recherche compatible avec le format Lucene.
- Internationalisation et localisation.
- Création de formulaires.
- Technologies de gestion des identités comme Microsoft InfoCard et OpenID.

12 http://framework.zend.com/

- Gestion de différents formats de web services dont XML-RPC, REST et Google Gdata.
- Système flexible de cache avec possibilité de le stocker en mémoire ou dans le système de fichiers.
- Composant de log inspiré de log4j.
- Composant purement PHP pour lire, mettre à jour et créer des fichiers PDF.
- Gestion de JSON pour faciliter le développement AJAX.
- Composant pour utiliser et construire des flux de syndication RSS et Atom.
- Nombreux composants permettant d'utiliser très simplement des web services comme Amazon E-Commerce Service, Akismet, del.icio.us, Flickr, StrikeIron, Yahoo!, Audioscrobbler ou encore Simpy.

Nous étudierons plus particulièrement ce framework dans ce cours.

1.4 Comment trouver de l'aide ?

La communauté PHP est très vivante et fourmille de ressources pour aider les débutants ou ceux qui souhaitent se perfectionner.

Il existe énormément de ressources en anglais concernant PHP. Les ressources francophones sont nettement moins nombreuses. Par ailleurs, elles sont souvent de moins bonne qualité et plus anciennes. Dans la mesure du possible, essayez de ne pas vous limiter aux seules ressources françaises. Vous passeriez à côté de la majorité des ressources de qualité.

1.4.1 Documentation officielle

La première chose à faire avant de se lancer dans un développement ou en cas de difficulté est d'aller consulter la documentation officielle de PHP[13]. Celle-ci est disponible dans de nombreuses langues dont le français.

Une remarque cependant, la version anglaise étant celle de référence, les autres versions sont traduites à partir de celle-ci. La version française peut-être être moins à jour que la version anglaise.

La documentation PHP est très complète et couvre la quasi totalité des fonctions, classes et concepts disponibles dans ce langage. N'hésitez pas à la parcourir régulièrement car il y a fort à parier que la fonction que vous allez écrire existe déjà.

1.4.2 Articles et tutoriels

Il existe de nombreux sites traitant de PHP. En voici une sélection non exhaustive :

- Nexen.net : http://www.nexen.net/
- PHPDeveloper.org : http://www.phpdeveloper.org/
- Zend Developer Zone : http://devzone.zend.com/

1.4.3 Forums d'entraide

Voici une sélection de sites français proposant des forums d'entraide :

- PHPFrance : http://www.phpfrance.com/

13 http://www.php.net/manual/fr/

- PHP Débutant : http://www.phpdebutant.org/

Le principe même de ces sites est l'entraide. Poser une question, attendre la réponse et ne plus jamais revenir c'est s'assurer que le site mourra tôt ou tard.

Comme il y a toujours plus fort que soit, il y a également toujours moins fort. N'hésitez donc pas à aider ceux qui n'ont pas encore votre niveau. Vous verrez cela fait plaisir d'aider et cela peut vous apporter avec le temps une reconnaissance bienvenue.

1.5 Installer PHP

Installer PHP est relativement simple pour peu qu'on prenne le temps de se documenter un peu les premières fois.

Il faut bien noter que nous allons voir comment installer PHP sur un poste de développement. Ces méthodes d'installation ont pour but d'être simples. Elle ne sont pas optimisées ni sécurisées outre mesure. Il est donc fortement déconseillé d'installer PHP de cette manière sur un serveur de production.

1.5.1 Modes d'installation

PHP peut être servir pour trois utilisations différentes :

- créer des sites et des applications web
- créer des scripts en ligne de commande
- créer des applications graphiques

La première utilisation est la plus courante et c'est celle que nous allons développer dans la suite de ce cours. La seconde utilisation sera évoquée un peu plus loin. Enfin, la troisième reste encore très marginale et ne sera pas évoquée ici. Si le sujet vous intéresse, intéressez vous à l'extension PHP-GTK[14].

L'installation de PHP sur un serveur web nécessite une sorte de connecteur. C'est ce que l'on appelle des SAPI ou Server API. Il en existe des spécifiques à un serveur web comme pour Apache ou IIS et des génériques comme CGI[15] qui peut être utilisé avec de nombreux serveur web comme Lighttpd.

Pour l'installation d'un poste de développement, nous allons voir comment installer PHP 5 en tant que module d'Apache 2 sur les principaux systèmes d'exploitation.

Si vous souhaitez plus de détails n'hésitez pas à consulter la documentation de PHP sur ce point[16].

1.5.2 Installation sur Windows

L'installation d'un poste de développement sur Windows est très simple grâce à WampServer[17]. Cet installateur regroupe Apache, MySQL et PHP dans leurs dernières versions.

Il faut tout d'abord télécharger la dernière version de WampServer sur la page http://www.wampserver.com/download.php#bottom.

Il suffit ensuite de lancer l'installateur et répondre aux différentes questions posées.

---

14 http://gtk.php.net/
15 http://en.wikipedia.org/wiki/Common_Gateway_Interface
16 http://www.php.net/manual/fr/install.php
17 http://www.wampserver.com/

1.5.3 Installation sur Mac OS X

Comme sous Windows, il existe un installateur PHP sous Mac OS X : MAMP[18].

Celui-ci peut-être télécharger sur la page suivante : http://www.mamp.info/en/download.html.

Une fois le fichier DMG téléchargé, il suffit de glisser/déposer le fichier « MAMP » dans le répertoire « Applications » de votre système. Le fichier « MAMP PRO » est une application payante d'aide à la configuration de MAMP. Il n'est pas nécessaire pour faire fonctionner MAP et peut donc être ignoré.

1.5.4 Installation sur Unix / Linux

1.5.4.1 Installation assistée

L'installation de PHP sur un environnement Unix ou Linux est assez simple si l'on utilise les paquets précompilés de sa distribution. Si vous n'êtes pas familier avec la compilation de logiciels, c'est la meilleure manière de procéder.

Sur une distribution Debian, il suffit de lancer la commande suivante :

```
apt-get install libapache2-mod-php5
```

Ensuite, pour installer une extension PHP, il suffit d'appeler à nouveau la commande apt-get et de lui donner le nom de l'extension. Ainsi pour installer l'extension MySQL il faut utiliser la commande suivante :

```
apt-get install php5-mysql
```

Il est possible d'obtenir la liste des extensions disponibles avec cette commande :

```
apt-cache search php5-
```

Ces commandes sont à adapter si vous utilisez une distribution qui ne possède pas la commande apt-get comme Red Hat, Fedora ou encore Suse. Celles-ci utilisent une commande similaire appelée yum. Consultez sa documentation pour voir comme l'utiliser.

1.5.4.2 Installation manuelle

Que ce soit pour activer une extension très particulière ou dans le but de mieux comprendre le fonctionnement de PHP, il est possible de compiler PHP depuis le code source. Attention cependant, cela nécessite des connaissances avancées de la compilation d'application sous Unix.

Tout d'abord, il faut télécharger le code source de la dernière version stable de PHP.

TODO: Terminer de décrire l'installation sous UNIX / Linux

1.6 Configuration de PHP

PHP possède de nombreuses directives de configuration. Celles-ci permettent de modifier son comportement, d'activer des extensions et de les paramétrer.

Il existe quatre types de directives :

- PHP_INI_USER qui peuvent être définies dans les scripts PHP ou dans la base de registre de Windows,
- PHP_INI_PERDIR qui peuvent être définies dans le fichiers php.ini, httpd.conf et .htaccess,

18 http://www.mamp.info/en/mamp.html

- PHP_INI_SYSTEM qui peuvent être définies dans les fichiers php.ini et httpd.conf,
- PHP_INI_ALL qui peuvent être définies n'importe où.

Vous pouvez trouver les niveaux de configuration des différentes directives dans la documentation de PHP[19].

1.6.1 Méthodes de configuration

Il existe différentes méthodes pour modifier les directive de configuration de PHP.

1.6.1.1 Fichier php.ini

Le fichier php.ini est le fichier qui contient toutes les directives de configuration de PHP. Il est lu au chargement de PHP.

Après une modification de ce fichier, il faut donc ne pas oublier de recharger PHP ce qui signifie redémarrer le serveur HTTP si PHP est chargé en tant que module Apache par exemple.

Le fichier php.ini est situé a des endroits différents selon le système d'exploitation et le type d'installation choisi. Sur Unix / Linux, il est généralement situé dans le répertoire /etc/php5/apache2. Sur une installation Windows avec WampServer, celui-ci est situé dans le répertoire c:\wamp\bin\apache\apache2.x.x\bin.

Les directives situées dans le fichier php.ini s'appliquent à l'ensemble des scripts PHP du serveur sauf si elles sont surchargées par les niveaux supérieurs de configuration.

Voici un extrait du fichier php.ini :

```
[Date]
; Defines the default timezone used by the date functions
date.timezone = Europe/London

;date.default_latitude = 31.7667
;date.default_longitude = 35.2333
```

Comme vous pouvez le constater, le fichier php.ini est un fichier ini typique. Ainsi, la ligne 1 indique le début d'une section. La ligne 3 donne la valeur « Europe/Paris » à la directive de configuration « date.timezone » tandis que les autres lignes sont inactives car précédées d'un point-virgule qui indique un commentaire.

Enfin, pour activer une extension, il suffit d'ajouter une directive « extension » et lui donner pour valeur le nom de l'extension :

```
; Unix / Linux
extension=myql.so


; Windows
extension=mysql.dll
```

1.6.1.2 Fichier de configuration d'Apache

Apache permet de modifier les directives de configuration PHP dans ses fichiers de configuration

19 http://fr.php.net/manual/fr/ini.php#ini.list

(configuration principale ou VirtualHosts).

Voici un exemple de configuration de PHP adans un VirtualHost :

```
<VirtualHost *>
    ServerName www.domain.tld
    DocumentRoot /var/www/domain.tld/www/public
    <Directory /var/www/domain.tld/www/public>
        # Configuration d'Apache
        AllowOverride None
        Order allow,deny
        Allow from all

        # Configuration de PHP
        php_value date.timezone "Europe/Paris"
        php_flag display_errors off
        php_admin_value error_reporting 8191
        php_admin_flag upload_max_filesize "8M"
    </Directory>
</virtualHost>
```

La modification d'une directive de configuration dans un fichier de configuration d'Apache peut se faire de quatre manières différentes :

- **php_value** modifie la valeur de la directive spécifiée. Cette instruction n'est utilisable qu'avec les directives PHP de type PHP_INI_ALL et PHP_INI_PERDIR. Pour annuler une valeur qui aurait été modifiée au préalable, utilisez la valeur « none »,
- **php_flag** modifie la valeur de la directive spécifiée. Celle-ci doit être de type booléen. Cette instruction n'est utilisable qu'avec les directives PHP de type PHP_INI_ALL et PHP_INI_PERDIR.,
- **php_admin_value** qui est identique à php_value à ceci près qu'elle ne peut être utilisé dans un fichier .htaccess,
- **php_admin_flag** qui est identique à php_flag à ceci près qu'elle ne peut être utilisé dans un fichier .htaccess.

En modifiant une directive de configuration dans un VirtualHost, on peut définir une configuration spécifique pour un site précis.

1.6.1.1 Fichiers .htaccess

La syntaxe pour modifier une directive de configuration dans un fichier .htaccess est similaire à celle des VirtualHosts :

```
php_value date.timezone "Europe/Paris"
php_flag display_errors off
```

L'utilisation d'un fichier .htaccess pour définir des directives de configuration permet de n'appliquer celles-ci qu'à un répertoire et ses sous-répertoires.

A noter qu'il est impossible d'utiliser les instructions php_admin_value et php_admin_flag dans un

fichier .htaccess.

1.6.1.2 Scripts PHP

La couche de configuration la plus haute est située directement dans les scripts PHP. La fonction ini_set() permet de définir la valeur d'une directive de configuration :

```
<?php
ini_set('date.timezone', 'Europe/Paris');
?>
```

Il est possible de récupérer la valeur d'une directive de configuration avec la fonction ini_get().

1.6.1.3 Base de registre Windows

Lorsque vous utilisez PHP sur Windows, la configuration peut être modifiée pour chaque répertoire en utilisant la base de registre de Windows. Les valeurs de configuration sont stockées avec la clé de registre HKLM\SOFTWARE\PHP\Per Directory Values, dans les sous-clés correspondantes aux noms des répertoires.

Par exemple, la valeur d'une option dans le répertoire c:\wamp\www sera stockée dans la clé HKLM\SOFTWARE\PHP\Per Directory Values\c\wamp\www. La valeur de cette option sera utilisée pour tous les scripts qui fonctionnent dans ce répertoire ou ses sous-répertoires.

Les valeurs sous la clé doivent avoir le nom d'une direction de configuration PHP, et la valeur correspondante. Les constantes PHP ne sont pas utilisables : il faut mettre la valeur entière.

Attention toutefois, seules les valeurs des configurations dans PHP_INI_USER peuvent être fixées de cette manière, celles dans PHP_INI_PERDIR ne peuvent l'être.

1.6.2 Principales directives de configuration

Il existe de très nombreuses directives de configuration. Nous allons étudier quelques unes d'entre elles parmi les plus utiles. Reportez-vous à la documentation officielle[20] pour découvrir les autres.

1.6.2.1 max_execution_time

Cette directive définit en secondes la durée maximale d'exécution d'un script PHP. Par défaut celle-ci est de 30 secondes.

Afin d'éviter qu'un script mal écrit tourne indéfiniment utilisant plus de ressources que nécessaire, PHP arrête automatiquement les scripts dépassant cette durée d'exécution. L'arrêt est brutal et déclenche un message d'erreur.

1.6.2.2 memory_limit

PHP est conçu pour être utilisé sur des serveurs web qui traitent simultanément de nombreuses requêtes. Pour éviter qu'un script n'utilise toute la mémoire disponible, empêchant l'exécution des autres requêtes, un quota est alloué à chaque processus PHP. Cette directive permet de configurer cette valeur.

Comme pour toutes les directives qui demandent une taille, il est possible de spécifier la valeur de différentes manières :

---

20 http://fr.php.net/manual/fr/ini.php#ini.list

```
ini_set('memory_limit', 67108864); // En octets
ini_set('memory_limit', '65536K'); // En kilooctets
ini_set('memory_limit', '128M');   // En megaoctets
ini_set('memory_limit', '1G');     // En gigaoctets
```

A noter, qu'en donnant une valeur de -1 à cette directive, on leve la limite mais attention à ne garder cette utilisation que pour des cas très particuliers car cela risque fort de poser des problèmes.

1.6.2.3 error_reporting

Nous verrons plus en détail la gestion des des erreurs en PHP dans un prochain chapitre mais retenez pour le moment que PHP possède une hiérarchie d'erreurs. Celles-ci peuvent être déclenchée par PHP lui-même mais également par vos scripts. Les erreurs servent à signaler un problème lors de l'exécution d'un script.

Cette directive est l'une des plus importante car elle détermine à partir de quel niveau les erreurs ne seront plus ignorées. Ce paramètre est un entier, représentant un champ de bits. Pour définir sa valeur on peut utiliser constantes suivantes ou leurs valeurs entières :

- E_ERROR : 1
- E_WARNING : 2
- E_PARSE : 4
- E_NOTICE : 8
- E_CORE_ERROR : 16
- E_CORE_WARNING : 32
- E_COMPILE_ERROR : 64
- E_COMPILE_WARNING : 128
- E_USER_ERROR : 256
- E_USER_WARNING : 512
- E_USER_NOTICE : 1024
- E_STRICT : 2048
- E_RECOVERABLE_ERROR : 4096
- E_ALL : 6143

La configuration par défaut est E_ALL & ~E_NOTICE. Elle montre toutes les erreurs, sauf les E_NOTICE.

Pendant le développement, et de manière plus générale pour un code de qualité, il est recommandé de donner la valeur E_ALL | E_STRICT à cette directive afin de voir l'ensemble des erreurs relevées par PHP.

Enfin, les constantes ne peuvent être utilisées que dans le fichier php.ini ou avec la fonction ini_set(). Dans les autres méthodes de configuration, il faut utiliser les valeurs entières.

1.6.2.4 display_errors

Cette directive est trop souvent ignorée. Son but est d'activer l'affichage des erreurs. L'erreur classique lorsqu'on ne veux pas afficher les erreurs, sur un serveur de production par exemple, est de

mettre la valeur « 0 » à la directive « error_reporting ».

En apparence le but est atteint mais il y a un effet de bord[21] important à ce comportement. Au lieu de ne simplement pas afficher les messages d'erreurs, PHP va définir le niveau d'erreur à 0 et ne va donc jamais réagir à aucune erreur sauf pour terminer un script provoquant une erreur fatale. Une éventuelle gestion des erreurs devient dans ce cas inopérante.

La manière correcte de cacher les messages d'erreur est tout simplement d'utiliser la directive « display_errors » en lui donnant la valeur « 0 ».

1.6.2.5 log_errors

Cette directive est complémentaire de « display_errors ». Elle permet de garder une trace dans les logs des messages d'erreur. Ainsi, sur un serveur de production il est recommandé de définir la directive « display_errors » à « 0 » et « log_errors » à « 1 ». Sur un poste de développement, c'est l'inverse qui est préconisé.

1.6.2.6 register_globals

Cette directive a pour valeur par défaut « 0 » depuis la version 5 de PHP et il est fortement déconseillé de la modifier. Nous verrons dans un chapitre ultérieur pourquoi.

1.6.2.7 magic_quote_gpc

PHP propose un mécanisme d'échappement des guillemets afin de protéger les scripts des injections SQL. Le but est louable mais le mécanisme est simpliste et pose plus de problèmes qu'il n'en résout.

Il est donc conseillé de mettre cette directive à « 0 ». Nous verrons dans un chapitre ultérieur comment se prémunir des injections SQL de manière plus efficace.

1.6.2.8 include_path

Lorsqu'on tente d'inclure un script PHP dans un autre avec les fonctions include() ou require(), celui-ci cherche par défaut le script dans le répertoire courant.

Avec la directive « include_path » il est possible de définir un ensemble de chemins systèmes dans lesquels PHP ira successivement chercher les fichiers qu'il devra inclure. Ces chemins sont séparés par un caractère dépendant du système d'exploitation, « : » sur Unix et « ; » sur Windows. A noter que la constante « PATH_SEPARATOR » contient le bon séparateur pour le système d'exploitation sur lequel le script est exécuté.

```
<?php
ini_set('include_path', '/var/www/library' . PATH_SEPARATOR . '.');
?>
```

Enfin, la valeur « . » équivaut aux répertoire courant.

21 http://fr.wikipedia.org/wiki/Effet_de_bord_(informatique)

2 Les structures de base

Dans ce chapitre, nous allons voir les fondements de la programmation PHP.

2.1 Écriture du code PHP

Le code PHP est contenu dans des scripts dont l'extension est généralement .php. Celle-ci peut cependant varier selon la configuration du serveur. Ainsi, il arrive de voir des scripts PHP avec les extensions .php4 ou .php5.

PHP est un langage de script qui peut être inséré directement dans du code HTML. Il peut cependant également être stocké dans un script sans HTML.

Voici un exemple de code PHP :

```
<html>
    <head>
        <title>Exemple</title>
    </head>
    <body>
        <h1>Titre de la page</h1>
        <?php
            echo '<p>Un texte ajouté avec PHP</p>';
        ?>
    </body>
</html>
```

2.1.1 Balises PHP

Le code PHP est encadré d'une balise ouvrante et d'une balise fermante. Il existe plusieurs manières d'écrire ces balises :

```
<html>
    <head>
        <title>Exemple</title>
    </head>
    <body>
        <h1>Titre de la page</h1>
        <?php echo '<p>Voici la méthode classique</p>'; ?>

        <? echo '<p>Voici la méthode avec des balises courtes</p>'; ?>

        <% echo '<p>Voici la méthode inspirée d'ASP</p>'; %>

        <script language="php">
            echo '<p>Voici une méthode alternative</p>';
        </script>
```

```
        <?='<p>Voici une méthode très courte</p>'?>
    </body>
</html>
```

La première méthode est fortement recommandée car c'est la seule qui ne soit pas désactivable dans la configuration. Les scripts ainsi écrits seront donc reconnus partout.

A noter dans ce cas que la balise fermante n'est pas obligatoire. Elle peut être implicite comme nous le verrons dans le chapitre consacré aux bonnes pratiques de développement.

2.1.2 Instructions

Une instruction PHP est un ordre que donne le script à PHP afin que celui-ci effectue une tâche. Une instruction est obligatoirement terminée par un point-virgule. Les espaces et retours à la ligne ne sont pas pris en compte. On peut ainsi mettre plusieurs instructions sur une même ligne ou répartir une instruction sur plusieurs lignes.

```php
<?php
// Une instruction par ligne
$a = 2;
$b = 3;
$c = $a + $b;
echo $c;

// Plusieurs instructions sur la mÃªme ligne
$a = 2; $b = 3; $c = $a + $b; echo $c;

// Une instruction repartie sur plusieurs lignes
$a
=
2;
$b
=
3;
$c
=
$a + $b;
echo $c;
?>
```

C'est au développeur de choisir l'écriture la plus adaptée pour assurer la lisibilité du code source.

Attention, l'oubli du point virgule après une instruction entraînera une erreur d'interprétation de la part de PHP.

2.1.3 Exécuter du code PHP

Le code PHP peut être exécuté de différentes manière selon l'environnement sur lequel il est déployé. L'usage le plus courant est celui sur le web avec uns erveur HTTP comme Apache, par exemple.

Dans ce cas, il suffit de mettre le script PHP dans un répertoire situé sous la racine d'Apache et d'appeler ce script depuis votre navigateur. Cela se fait généralement en appelant l'URL http://localhost/nom_du_script.php.

2.1.4 Commentaires

Il est possible d'écrire des commentaires dans un script PHP. Ce sont des lignes qui ne seront pas

interprétées par PHP et qui permettent à un développeur de laisser une note dans un script expliquant par exemple son fonctionnement.

Il est possible d'écrire des commentaires de plusieurs manières :

```php
<?php
// Ceci est un commentaire sur une ligne

/*
 Ceci est un commentaire
 sur plusieurs
 lignes
*/

# Ceci est un commentaire de type shell
?>
```

2.2 Variables et constantes

2.2.1 Variables

Les variables sont une sorte de récipients qui contiennent des valeurs. Ces variables peuvent être définies, modifiées, utilisées et supprimées au cours du traitement d'un script.

Les variables PHP doivent suivre un certain nombre de règles :

- elles commencent par le symbole « $ »
- elle ne peuvent pas contenir des espaces ni les symboles « - », « @ » et « . »
- elle ne peuvent pas commencer par un chiffre

Pour le reste, vous êtes libres de nommer vos variables comme bon vous semble.

2.2.1.1 Utilisation des variables

```php
<?php
// Définition de la variable
$variable = 'Une variable contenant du texte';

// Modification de la valeur de la variable
$variable = 'Nouvelle valeur';

// Affichage de la variable
echo $variable;

// Suppression de la variable
unset($variable);
?>
```

2.2.1.2 Portée des variables

Les variables ont une portée. C'est à dire qu'une variable une fois qu'elle est définie n'est pas disponible partout dans le script PHP. De même, une variable définie dans un script ne sera pas disponible dans un autre script. Enfin, chaque appel à un script possède des variables propres dont il ne partage pas les valeurs avec les autres appels.

Il existe deux portée de variables en PHP : la portée locale et la portée globale.

La portée locale fait qu'une variable ne sera disponible que dans le bloc de code où elle a été définie. Une fonction ou une méthode de classe sont des exemples de bloc de code.

```php
<?php
// Les deux variables $a ne sont pas dans la même portée.
// Il s'agit donc de deux variables différentes.
function test()
{
    $a = 5;
}

$a = 2;
$b = 7;
?>
```

Pour avoir accès à la variable $b dans la fonction test(), il faudrait passer la valeur de $b comme un argument de la fonction.

La portée globale est en fait un bloc de code un peu particulier. Il s'agit de celui du script en cours d'exécution. Ainsi dans l'exemple ci-dessus, la variable $a qui vaut 2 est dans la portée globale. Cette portée à la particularité qu'il est possible d'accéder à ses variables depuis les portées locales. Cela peut se faire de deux manières :

- en utilisant la variable spéciale $GLOBALS qui est un tableau contenant toutes les variables de la portée globale, dites variables globales;
- en utilisant le mot clé « global » dans une portée locale.

```php
<?php
function test()
{
    // Permet de rendre la variable globale $a disponible dans cette fonction
    global $a;

    return $a + 3;
}

$a = 12;
$c = test();

// Affiche 15
echo $c;
?>
```

2.2.1.3 Obtenir des informations sur une variable

La fonction « var_dump() » permet de connaître le type et la valeur d'une ou plusieurs variables.

```php
<?php
$a = 13;
$b = 'orange';

// Affiche les informations concernant une variable
var_dump($a);

// Affiche les informations concernant plusieurs variables
var_dump($a, $b);
?>
```

2.2.2 Constantes

Les constantes sont assez similaires aux variables à la différence près qu'une fois définie, leur valeur ne peut plus être changée. Pour définir une constante, on utilise la fonction define().

Une autre différence est que le nom d'une constante ne commence pas par le symbole $. Enfin, par convention les constantes sont écrites en majuscules.

```php
<?php
// Définition de la constante
define('MA_CONSTANTE', 'La valeur de ma constante');

// Affichage de la constante
echo MA_CONSTANTE;
?>
```

2.3 Types de données

PHP est un langage faiblement typé, c'est à dire qu'il n'est pas nécessaire de spécifier le type d'un variable lors de sa déclaration. PHP détermine automatiquement le type de la variable selon sa valeur. Ce type peut d'ailleurs changer au cours du script si la valeur est modifiée.

PHP supporte huit types basiques répartis en trois catégories :

- 4 types scalaires :
  - boolean (booléen)
  - integer (entier)
  - float (nombre décimaul)
  - string (chaîne de caractères)
- 2 types composés :
  - array (tableau)

    - object (objet)
- Ainsi que 2 types spéciaux :
    - resource (ressource)
    - null

2.3.1 Booléen

Le type booléen peut prendre uniquement deux valeurs « true » et « false » selon que la valeur est vraie ou fausse.

A noter que les valeurs de ce type ne tiennent pas compte de la casse. Aussi, il est possible d'utilsier les valeurs « TRUE » et « FALSE ».

2.3.2 Entier

Les nombres entiers, qu'ils soient positifs ou négatifs, peuvent être utilisé directement comme valeur d'une variable.

Par défaut les nombres entiers sont exprimés en base 10. Il est possible d'utiliser la notation octale (base 8) en faisant précéder la valeur d'un 0. Il est également possible d'utiliser la notation hexadécimale (base 16) en faisant précéder la valeur de « 0x ».

```php
<?php
// Ces trois variables ont pour valeur 12

$a = 12;   // Notation décimale
$b = 014;  // Notation octale
$c = 0x0C; // Notation hexadécimale
?>
```

2.3.3 Nombre décimal

Les nombres décimaux sont également appelés « nombres à virgule flottante », « nombres réels », « floats » ou encore « doubles ».

Il est possible d'utiliser la notation simple ou la notation scientifique :

```php
<?php
$a = 4.456; // 4.456
$b = 6.3e5; // 630000
$c = 2e-3;  // 0.002
?>
```

2.3.4 Chaîne de caractère

Une chaîne de caractères est une suite de caractères. Il existe trois syntaxes différentes pour déclarer une chaîne de caractères.

```php
<?php
// Syntaxe avec guillemets simples
$a = 'Une chaîne de caractères';
```

```
// Syntaxe avec guillemets doubles
$b = "Une chaîne de caractères";

// Syntaxe Heredoc
$c = <<<EOT
Une chaîne de caractères
qui s'étale
sur plusieurs lignes
EOT;
?>
```

Nous verrons ce type plus en détails dans un prochain chapitre.

2.3.5 Tableau

Un tableau est un type composé qui contient des paires de clés et de valeurs associées. Il en existe deux sortes : les tableaux indexés et les tableaux associatifs.

```
<?php
// Tableau indexé
$a = array('pomme', 'fraise', 'orange');

// Tableau associatif
$b = array(
    'pomme'  => 1.35,
    'fraise' => 2.40,
    'orange' => 1.75,
)
?>
```

Nous verrons ce type plus en détails dans un prochain chapitre.

2.3.6 Objet

Un objet est une instance, c'est à dire une représentation concrète, d'une classe. Il possède des méthodes et des propriétés. C'est un concept de base de la programmation orientée objet[22] (POO) que nous étudierons dans un prochain chapitre.

```
<?php
class MaClasse
{
}

$objet = new MaClasse();
?>
```

2.3.7 Ressource

Une ressource est une variable spéciale, contenant une référence vers une ressource externe. C'est le seul type que dont on ne peut pas créer une valeur. On ne peut que manipuler les ressources créées par certaines fonctions de PHP.

22 http://fr.wikipedia.org/wiki/Poo

Une ressource peut être un fichier ouvert, une connexion à une base de données ou une image par exemple.

2.3.8 Null

La type null est un type très spécial car il indique qu'il n'y a pas de valeur pour la variable. La seule valeur possible pour ce type est « null ».

A noter que comme pour le type booléen, le type null ne tient pas compte de la casse. On peut donc utiliser les valeurs « null » et « NULL ».

2.3.9 Vérification du type

Bien que PHP soit un langage faiblement typé, il peut-être utile dans certains cas de s'assurer du type d'une variable.

Il existe une série de fonctions pour vérifier qu'une variable est bien d'un type précis.

Fonction	Description
is_array()	Détermine si une variable est un tableau
is_binary()	Vérifie si une variable est une chaîne de caractères binaire native
is_bool()	Détermine si une variable est un booléen
is_double()	Alias de is_float()
is_float()	Détermine si une variable est de type nombre décimal
is_int()	Détermine si une variable est de type nombre entier
is_integer()	Alias de is_int()
is_long()	Alias de is_int()
is_null()	Indique si une variable vaut null
is_numeric()	Détermine si une variable est un type numérique
is_object()	Détermine si une variable est de type objet
is_real()	Alias de is_float()
is_resource()	Détermine si une variable est une ressource
is_scalar()	Indique si une variable est un scalaire
is_string()	Détermine si une variable est de type chaîne de caractères

2.3.10 Transtypage

La plupart du temps le type d'une variable importe peu. Seule sa valeur est importante. Cependant, il est parfois utile de modifier le type d'une variable. Cette opération s'appelle le transtypage.

PHP effectue parfois un transtypage implicite lors d'opérations comme la multiplication de deux chaînes de caractères contenant des entiers ou la concaténation d'un entier et d'une chaîne de caractères.

Le transtypage s'effectue faisant précéder la variable du nouveau type entre parenthèses. Le tableau suivant énumère les différentes possibilités.

Expression	Description
(int), (integer)	Convertit en entier
(bool), (boolean)	Convertit en booléen
(float), (double), (real)	Convertit en nombre décimal
(string)	Convertit en chaîne de caractères
(array)	Convertit en tableau
(object)	Convertit en objet

```php
<?php
$a = 3;
var_dump($a); // Affiche int(3)

$b = (string) $a;
var_dump($b); // Affiche string(1) "3"
?>
```

Dans certains cas, la valeur de la variable peut être modifiée lors du transtypage. Celle-ci dépend du type vers lequel on convertit la variable.

La plupart du temps la conversion se fait de manière naturelle mais il existe de multiples cas particuliers répertoriés par la documentation de PHP[23].

23 http://fr.php.net/manual/fr/language.types.type-juggling.php

3 Les traitements de base

Nous allons à présent voir comment manipuler les structures vues au chapitre précédent.

3.1 Opérateurs

Les opérateurs sont des symboles qui permettent de manipuler des variables et des constantes. Il en existe de plusieurs sortes comme nous allons le voir.

3.1.1 Opérateurs d'assignation

Comme leur nom l'indique, les opérateurs d'affectation permettent d'affecter une valeur à une variable.

L'opérateur « = » est l'opérateur d'assignation le plus courant. Il donne la valeur située à sa droite à la variable située à sa gauche.

```
<?php
// Assigne la valeur 5 à la variable $a
$a = 5;
?>
```

Attention, contrairement à ce que l'on pourrait penser l'opérateur « = » n'indique pas une comparaison entre la valeur de la variable $a et le chiffre 5. Pour cela il faudrait utiliser l'opérateur de comparaison « == » que nous verrons un peu plus loin.

Lorsque l'on assigne la valeur d'une variable à une autre, cette assignation peut se faire par valeur ou par référence. Prenons un exemple pour illustrer la différence :

```
<?php
// Assignation par valeur
$a = 5;
$b = 7;
$a = $b; // $a et $b valent 7
$b = 8;  // $a vaut 7 et $b vaut 8

// Assignation par référence
$a = 5;
$b = 7;
$a = &$b; // $a et $b valent 7
$b = 8;  // $a et $b valent 8
$a = 2;   // $a et $b valent 2
?>
```

Comme on peut le voir, l'assignation par valeur fonctionne de manière assez naturelle. L'assignation par référence, en revanche, demande plus d'explication.

Tout d'abord, pour indiquer une affectation on utilise le symbole « & ». Cela a pour effet de lier les deux variables. Ainsi lorsque l'une d'elle change de valeur, l'autre également.

Pour faire une petite parenthèse technique, en interne, PHP ne gère qu'une seule valeur et fait

pointer les deux variables vers celle-ci.

3.1.2 Opérateurs arithmétiques

Il existe plusieurs opérateurs arithmétiques. Tout d'abord, on trouve les opérateurs simples comme « + », « - », « * » et « / » qui correspondent respectivement à l'addition, la soustraction, la multiplication et la division.

```
<?php
$a = 3 + 5;   // $a vaut 8
$b = $a - 4;  // $b vaut 4
$c = $a * $b; // $c vaut 32
$d = $c / 4;  // $d vaut 8
?>
```

Il existe également un autre opérateur arithmétique nommé modulo et représenté par le symbole « % ». Il renvoie le reste de la division de la première valeur par la seconde.

```
<?php
$e = $a % 3;  // $e vaut 3
?>
```

A noter que les opérandes du modulo sont converties en entiers (en supprimant la partie décimale) avant exécution.

Il est également possible d'incrémenter ou de décrémenter la valeur d'une variable. Cela revient à renvoyer la valeur de la variable puis à en augmenter ou diminuer la valeur de un. On parle dans ce cas de post-décrémentation et de post-décrémentation.

```
<?php
$a++; // Équivaut à "$a = $a + 1;"
$b--; // Équivaut à "$b = $b - 1;"
?>
```

PHP supporte également les opérateurs de pre-incrémentation et pre-décrémentation, comme en langage C. La différence se situe dans le fait que l'incrémentation ou la décrémentation s'effectue avant le renvoi de la valeur.

```
<?php
$a = 3;
echo $a++; // Affiche 3
echo $a;   // Affiche 4
echo --$a; // Affiche 3
?>
```

3.1.3 Opérateurs de comparaison

Comme leur nom l'indique, les opérateurs de comparaison permettent de comparer deux valeurs. Leur évaluation renvoie une valeur booléenne, « true » ou « false » donc.

Voici un tableau qui liste les différents opérateurs :

Nom	Exemple
Égal à	$a == $b
Identique à	$a === $b
Différent de	$a != $b
Différent de	$a <> $b
Différent de	$a !== $b
Inférieur à	$a < $b
Supérieur à	$a > $b
Inférieur ou égal à	$a <= $b
Supérieur ou égal à	$a >= $b

Les opérateurs « === » et « !== » ont un comportement un peu particulier. En plus de comparer les valeurs des opérandes, ils comparent leurs types. Ainsi, « === » ne renvoie « true » que si les valeurs et les types sont identiques. Quant à l'opérateur « !== », il ne renvoie « true » que si au moins les valeurs ou les types sont différents.

Une erreur courante consiste à utiliser le symbole « = » pour comparer deux valeurs. Cette opération renverra toujours « true » car il ne s'agit pas d'une comparaison mais d'une assignation de la valeur de droite à celle de gauche.

Enfin, notez que si vous comparez un entier avec une chaîne, la chaîne est convertie en un nombre. Si vous comparez deux chaînes numériques, elles seront comparées en tant qu'entiers.

3.1.4 Opérateurs logiques

TODO: Rédiger cette partie

3.1.5 Opérateurs sur les bits

Les opérateurs sur les bits vous permettent de manipuler les bits dans un entier. Si les paramètres de gauche et de droite sont des chaînes de caractères, l'opérateur de bits agira sur les valeurs ASCII de ces caractères.

Voici un tableau qui liste les différents opérateurs :

Nom	Exemple	Description
Et (and)	$a & $b	Les bits positionnés à 1 dans $a ET dans $b sont positionnés à 1
Ou (or)	$a | $b	Les bits positionnés à 1 dans $a OU $b sont positionnés à 1
Ou exclusif (xor)	$a ^ $b	Les bits positionnés à 1 dans $a OU dans $b mais pas dans les deux sont positionnés à 1
Non (not)	~ $a	Les bits qui sont positionnés à 1 dans $a sont positionnés à 0, et vice versa

Décalage à gauche	$a << $b	Décale les bits de $a, $b fois sur la gauche (chaque décalage équivaut à une multiplication par 2)
Décalage à droite	$a >> $b	Décalage des bits de $a, $b fois par la droite (chaque décalage équivaut à une division par 2)

3.1.6 Opérateur de concaténation

La concaténation est l'opération de rassemblement de plusieurs chaînes de caractères en une seule. L'opérateur de concaténation est représenté par le symbole « . ».

```php
<?php
$a = 'bleu';
echo 'Le ciel est ' . $a; // Affiche "Le ciel est bleu"
?>
```

3.1.7 Opérateurs combinés

Il est possible de combiner les opérateurs d'assignation avec les opérateurs arithmétiques ou l'opérateur de concaténation. Cela permet d'utiliser la valeur d'une variable dans une expression et d'affecter le résultat de cette expression à cette variable.

```php
<?php
$a  = 6;
$a += 3; // Équivaut à "$a = $a + 3;" soit 9

$b  = 'Le ciel est ';
$b .= 'bleu'; // $b vaut "Le ciel est bleu"
?>
```

3.1.8 Précédence des opérateurs

La précédence, ou priorité, des opérateurs indique l'ordre dans lequel les valeurs doivent être analysées.

Par exemple, dans l'expression 4 + 5 * 3, le résultat est 19 et non 27, car la multiplication a une priorité supérieure par rapport à l'addition. Des parenthèses peuvent être utilisées pour forcer la priorité, si nécessaire. Par exemple : (4 + 5) * 3 donnera 27. Si la priorité d'opérateur est égale, l'associativité de gauche à droite est utilisée.

Le tableau suivant dresse une liste de la précédence des différents opérateurs dans un ordre décroissant de priorité. Les opérateurs sur une même ligne ont une priorité équivalente et, dans ce cas, leur association détermine leur ordre d'évaluation.

Opérateurs	Associativité
new	non-associatif
[	gauche
++ --	non-associatif

~ - (int) (float) (string) (array) (object) (bool) @	non-associatif
instanceof	non-associatif
!	droite
* / %	gauche
+ - .	gauche
<< >>	gauche
< <= > >= <>	non-associatif
== != === !==	non-associatif
&	gauche
^	gauche
|	gauche
&&	gauche
||	gauche
? :	gauche
= += -= *= /= .= %= &= |= ^= <<= >>=	droite
and	gauche
xor	gauche
or	gauche
,	gauche

3.2 Structures de contrôle

Les structures de contrôle permettent de répéter des actions ou bien de mettre une condition à leur exécution. Cette condition est généralement évaluée sous la forme d'un booléen dont la valeur va déterminer si la condition est remplie ou non.

Les structures de contrôle de PHP sont assez similaires à celles du langage C.

3.2.1 Les conditions

3.2.1.1 if

L'instruction « if » permet de soumettre l'exécution d'un bloc de code à une condition. Celle-ci sera évaluée afin d'en tirer un booléen. Si ce dernier vaut « true » le bloc de code sera exécuté et il ne le sera pas dans le cas contraire.

```php
<?php
$a = 3;

if ($a > 1) {
    // Ce bloc de code sera exécuté
```

```
}
?>
```

3.2.1.2 else

L'instruction « else » s'utilise en complément de l'instruction « if ». Elle permet d'exécuter un bloc de code alternatif au cas où la condition du « if » échouerait.

```
<?php
$a = 0;

if ($a > 1) {
    // Ce bloc de code ne sera pas exécuté
} else {
    // Ce bloc de code sera exécuté
}
?>
```

3.2.1.3 elseif

Comme son nom l'indique, l'instruction « elseif » combine les instructions « else » et « if ». Ainsi, le bloc de code qui la suit ne sera exécuté que si les conditions précédentes ont échoué et si la condition est remplie.

```
<?php
$a = 0;

if ($a > 1) {
    // Ce bloc de code ne sera pas exécuté
} elseif($a == 0) {
    // Ce bloc de code sera exécuté
} else {
    // Ce bloc de code ne sera pas exécuté
}
?>
```

3.2.1.4 switch

L'instruction switch permet d'exécuter différents blocs de code selon une valeur.

```
<?php
$a = 5;

switch($a) {
    case 1:
        // Bloc de code non exécuté
        break;
    case 2:
```

```
        // Bloc de code non exécuté
        break;
    case 3:
    case 4:
        // Bloc de code non exécuté
        break;
    case 5:
        // Bloc de code exécuté
        break;
    default:
        // Bloc de code non exécuté
        break;
}
?>
```

Comme vous pouvez le constater, l'instruction « switch » est plus compliquée que les instructions précédentes.

L'exemple ci-dessus vérifie la valeur de la variable « $a ». Les lignes « case X: » permettent de définir un bloc de code à exécuter si « $a » vaut X.

Vous remarquerez également qu'il est possible de coupler ces lignes « case » pour le cas où un même bloc de code doit être exécuté pour différentes valeurs de la variable comme c'est le cas ici pour les valeurs 3 et 4.

L'instruction « break » permet d'indiquer la fin d'un bloc de code et évite de coupler des lignes « case » par erreur.

Enfin, l'instruction « default » indique le bloc de code à exécuter au cas où aucune des lignes « case » n'aurait correspondu à la valeur de la variable. Cette instruction est facultative. Dans ce cas et si aucun « case » ne correspond, aucun bloc de code ne sera exécuté.

3.2.2 Boucles

Les boucles sont des instructions qui permettent d'exécuter un bloc de code à plusieurs reprises tout en contrôlant ces exécutions. Chaque tour d'une boucle est appelée « itération ». Enfin, il est possible d'imbriquer des boucles.

3.2.2.1 While

La boucle « while » est probablement le moyen le plus simple de faire une boucle en PHP.

Tant que la condition est évaluée à « true », le bloc de code correspondant sera exécuté encore et encore.

```
<?php
$a = 0;

while($a < 5) {
    echo '$a vaut : ' . $a;
    $a++;
```

```
}
?>
```

Si la condition n'est pas remplie avant la première itération, le bloc de code ne sera jamais exécuté.

Si l'évaluation de la condition passe à « false » lors d'une itération, celle-ci se poursuit jusqu'à son terme. La boucle s'arrêtera avant le début de la prochaine itération.

Enfin, veillez à faire évoluer les éléments de la condition dans la boucle sous peine de créer une boucle infinie.

3.2.2.2 do/while

La boucle « do/while » est assez similaire à la boucle « while » à cela près que la condition est évaluée à la fin de la boucle et non au début. Cela signifie que le bloc de code est au moins exécuté une fois.

```
<?php
$a = 0;

do {
    echo '$a vaut : ' . $a;
    $a++;
} while($a < 5)
?>
```

3.2.2.3 for

La boucle « for » est probablement l'une des plus utilisée. Elle fonctionne exactement comme en langage C.

```
<?php
for($a = 0; $a < 5; $a++) {
    echo '$a vaut : ' . $a;
}
?>
```

La partie située entre les parenthèses après le « for » est composée comme suit :

- La première partie est exécutée une seule fois avant le début de la boucle. Elle permet l'initialisation de variables. On en initialise généralement qu'une seule mais il est possible d'en initialiser plusieurs en les séparant par des virgules;
- La seconde partie est la condition qui conditionne la sortie de la boucle;
- Enfin, la troisième partie est exécuté à chaque itération de la boucle. Elle permet de faire évoluer les valeurs des variables.

3.2.2.4 foreach

La boucle « foreach » permet de parcourir simplement des tableaux ou des objets.

Elle possède deux syntaxes présentées dans l'exemple ci-dessous.

```
<?php
```

```php
$departements = array(1 => 'Ain', 2 => 'Aisne', 3 => 'Allier');

// Affiche
//
//  Ain
//  Aisne
//  Allier
foreach($departements as $departement) {
    echo $departement;
}

// Affiche
//
//  1 : Ain
//  2 : Aisne
//  3 : Allier
foreach($departements as $numero => $nom) {
    echo $numero . ' : ' . $nom;
}
?>
```

Dans la première syntaxe, à chaque itération, la variable « $departement » reçoit la valeur de l'élément courant du tableau. Dans la seconde syntaxe, la variable « $numero » reçoit la clé de l'élément courant du tableau et la variable « $nom » sa valeur.

La remise au début du pointeur du tableau est faite automatiquement, il n'est donc pas nécessaire d'appeler la fonction reset().

A noter enfin que, sauf si le tableau est passé par référence, l'instruction « foreach » travaille sur une copie du tableau.

3.2.2.5 break

L'instruction « break » est assez particulière. Elle permet de sortir d'une boucle sans attendre la fin des itérations. Elle est utilisable avec toutes sortes de boucles.

```php
<?php
// Permet de boucler en attendant qu'une condition devienne vraie
while(true) {
    if (/* condition */) {
        break;
    }
}
?>
```

Si plusieurs boucles sont imbriquées, il est possible de sortir de plusieurs de ces boucles avec une seule instruction « break ». Dans ce cas, il faut ajouter le nombre de boucle à quitter après « break ».

```php
<?php
```

```php
$i = 0;
while (++$i) {
    switch ($i) {
        case 5:
            // action
            break 1;  // Termine uniquement le switch
        case 10:
            // action
            break 2;  // Termine le switch et la boucle while
    }
}
?>
```

3.2.2.6 continue

L'instruction « continue » est assez similaire à l'instruction « break » sauf qu'au lieu de terminer une boucle, elle ne termine que l'itération courante. La boucle se poursuit donc mais en passant immédiatement à l'itération suivante.

De même que l'instruction « break », « switch » peut être suivant d'un nombre indiquant le nombre de boucles imbriquées impactées.

3.2.3 Instructions d'inclusion de fichiers

3.2.3.1 include

L'instruction «include» inclut et exécute un script PHP. Celui-ci remplace l'instruction dans le script appelant.

```php
<?php
include 'fichier.php';
include 'chemin/fichier.php';
?>
```

Les fichiers à inclure sont d'abord recherchés dans chaque dossier de « include_path », relativement au dossier courant, puis dans le dossier de travail du script. Si le nom du fichier commence par ./ ou ../, il est cherché uniquement dans le dossier courant d'exécution.

Si le fichier n'est pas trouvé, PHP émet une erreur de niveau E_WARNING.

Enfin, cette instruction étant une construction de langage et non une fonction, les parenthèses sont facultatives.

3.2.3.2 require

L'instruction « require » fonctionne exactement comme « include » excepté que si le fichier n'est pas trouvé, PHP émet une erreur de niveau E_ERROR qui stoppe l'exécution du script.

```php
<?php
require 'fichier.php';
require 'chemin/fichier.php';
```

```
?>
```

3.2.3.3 include_once / require_once

Comme leurs noms l'indiquent, les instructions « include_once » et « require_once » sont respectivement des versions des instructions « include » et « require » qui ne sont exécutées qu'une seule fois.

Cela est pratique pour éviter d'inclure plusieurs fois une déclaration de classe, ce qui aurait pour effet de provoquer une erreur fatale.

```php
<?php
include_once 'fichier.php';

// nombreuses lignes de code

// le fichier "fichier.php" ne sera pas inclus à nouveau
include_once 'fichier.php';
?>
```

A noter que, les informations concernant les fichiers déjà inclus sont partagées entre les instructions « include_once » et « require_once ».

3.3 Fonctions

3.3.1 Déclaration d'une fonction

Il existe deux types de fonctions dans le langage PHP : les fonctions internes et les fonctions définies par l'utilisateur. Les premières n'ont pas besoin d'être déclarées. Elle peuvent être utilisées n'importe quand et dans n'importe quel endroit d'un script.

En revanche, comme leur nom l'indique, les fonctions doivent être déclarées avant de pouvoir être utilisées. Étudions la structure d'une fonction :

```php
<?php
function test($argument1, $argument2, /* ..., */ $argumentn)
{
    echo 'Argument 1 : ' . $argument1;
    $resultat = $argument1 + $argument2;
    return $resultat;
}
?>
```

Une fonction possède un nom, ici « test », éventuellement des arguments et une valeur de retour.

Toutes les fonctions et classes en PHP ont une portée globale - elles peuvent être appelées à l'extérieur d'une fonction si elles ont été définies à l'intérieur et vice-versa.

PHP ne supporte pas la surcharge de fonction, ni la destruction ou la redéfinition de fonctions déjà déclarées.

3.3.1.1 Gestion des arguments

Il est possible de spécifier une valeur par défaut pour un argument. Dans ce cas, celui-ci peut être omis lors de l'appel de la fonction. Sa valeur sera alors la valeur par défaut.

Lors de l'appel d'une fonction, PHP vérifie la présence des arguments sans valeur par défaut. A noter qu'il est possible de passer plus d'arguments que demandé par la fonction.

Cette possibilité permet de déclarer des fonctions sans arguments mais qui gèrent en réalité un nombre variable d'arguments. Dans ce cas, on récupère les valeurs des arguments à l'aide de la fonction func_get_args().

```php
<?php
function test()
{
    $arguments       = func_get_args();
    $nombreArguments = count($arguments);

    for ($i = 0; $i < $nombreArguments; $i++) {
        echo $arguments[$i] . '<br>';
    }
}

test(2, 'pomme', 34, 1367.34);
?>
```

3.3.1.2 Valeur de retour

La valeur de retour d'une fonction est définie à l'aide de l'instruction « return ». Cette dernière a également pour effet de terminer l'exécution de la fonction. Si cette instruction est omise, la valeur retournée est « null ».

```php
<?php
function test($a)
{
    return $a + 4;
}

echo test(2); // Affiche 6
?>
```

On trouve parfois, cette instruction écrite « return() ». Celle-ci étant une construction de langage et non une fonction, les parenthèses sont facultatives et leur utilisation ralentit légèrement le traitement par PHP. Il est donc conseillé de les omettre.

3.3.2 Appel d'une fonction

L'appel d'une fonction se fait simplement en utilisant son nom et en passant éventuellement quelques arguments.

```php
<?php
```

```php
function test()
{
    $arguments       = func_get_args();
    $nombreArguments = count($arguments);

    for ($i = 0; $i < $nombreArguments; $i++) {
        echo $arguments[$i] . '<br>';
    }
}

// Utilisation de la fonction test() précédemment déclarée
test(2, 'pomme', 34, 1367.34);

// Utilisation d'une fonction interne à PHP
$a = pow(2, 8);
?>
```