PHP : Conseils d'expert (Ce qu'il faut savoir pour bien utiliser le langage PHP pour des développements Web)

PHP, pourquoi faire ?

Toute page Web est avant tout constituée d'un script de code en langages HTML/CSS qui détermine à la fois le contenu et des ordres pour contrôler l'affichage.

Mais les langages HTML et CSS se limitent à **décrire** contenu et présentation des pages ; ce ne sont pas des langages de programmation gérant des variables, et ils ne permettent pas de personnaliser à la volée, dynamiquement, le contenu des pages Web.

Le langage de programmation **PHP vient donc en complément des langages du Web pour rendre variable le code HTML/CSS (et même JavaScript) d'une page Web selon le contexte.**

Le langage PHP permet de rendre dynamique les pages Web notamment en **accédant directement aux bases de données** de manière à personnaliser le contenu des pages Web ou en **réaction aux interactions de l'utilisateur via les menus, liens, boutons et formulaires des pages**HTML.

PHP, quelle version ?

Le langage PHP a beaucoup évolué depuis son origine.

L'orientation objet est assurée (en grande partie) depuis la version PHP5.

PHP6 n'a jamais vu le jour et PHP7 n'est qu'une amélioration du PHP5.

Actuellement, **les scripts PHP sont majoritairement écrits en PHP7.**

PHP : un langage de programmation complet, orienté objet et interprété

Tout comme JavaScript, PHP est un **langage de programmation complet** incluant les notions de **variables**, traitements **conditionnels** et **répétitifs**, ainsi qu'une**orientation Objet**.

Le code PHP s'écrit sous forme de **scripts interprétés par le serveur Web** afin d'adresser au navigateur de l'utilisateur du code HTML/CSS/JavaScript personnalisé. En conséquence, le code PHP 'ne sort pas du serveur' Web ; **le navigateur Web de l'utilisateur ne reçoit que du code standard HTML/CSS/JavaScript.**

Les scripts sont de préférence **très modulaires et stockés dans des fichiers distincts** du code HTML de la page.

Le code source des scripts PHP s'écrit à l'aide de tout **éditeur de texte**.

Pour que le serveur Web déclenche l'interprétation ligne à ligne du code PHP,**tout fichier contenant du script PHP doit avoir une extension particulière reconnue par le logiciel serveur Web** (typiquement *.php*).

**Conseil**

**Effectuez le découpage modulaire des scripts en vue de favoriser la maintenabilité et la réutilisabilité des modules**. La fusion des différents modules de code source nécessaires pour réaliser une page sera effectuée par le serveur Web avant interprétation (voir les commandes *include* et *require*)

**Privilégiez des modules 100% script PHP** ; toutefois certains scripts (comme les *vues*) contiennent du code PHP intimement entremêlé avec du code HTML.

PHP et l'orienté Objet

Même si le développement orienté Objet est effectivement opérationnel et régulièrement pratiqué en PHP, ce type d'écriture nécessite un code plus volumineux. Et ce code est à interpréter par le serveur Web qui doit déjà partager ses ressources entre tous les utilisateurs simultanés, ce qui peut conduire rapidement à une dégradation des performances et du service rendu.

**Conseil**

**Le développement orienté Objet systématique ne se justifie pas en PHP** ; il peut être judicieux de mixer l'écriture de scripts procéduraux et orientés Objet.

PHP : extensions et frameworks

Monde du 'logiciel libre' oblige, **le langage PHP est assez peu standardisé et comporte de très nombreuses instructions et fonctions** ; parfois, 2 mots-clés distincts exécutent exactement le même traitement ou des variantes infimes.

Il existe une**multitude d'extensions au langage** plus ou moins livrées selon les distributions de PHP.

Ces extensions sont en général matérialisées par de **nombreuses fonctions qui enrichissent le langage**.

**Conseil**

Avant de créer des (bibliothèques de) fonctions, le développeur a tout intérêt à rechercher et installer (ou simplement activer) une extension déjà opérationnelle.

Au-delà des nombreuses bibliothèques, **des frameworks complets (et souvent gratuits)** sont disponibles sur le Web ; ils offrent des modules de code (orienté objet) prêts à l'emploi et**imposent une structuration particulière du code PHP** (en général selon une architecture *MVC*).

L'utilisation d'un framework PHP nécessite un développement orienté objet et donc**l'instanciation et la manipulation de nombreux objets souvent complexes**.

Le code total à interpréter pour réaliser une page Web est bien plus volumineux que dans le cas d'une écriture 100% manuelle par le développeur.

**Conseil**

Avant d'utiliser un *framework* de développement PHP comme *Symfony*, *Zend* ou *Laravel*, il est indispensable de bien estimer la charge et la puissance du serveur Web nécessaires pour assurer le trafic attendu.

La syntaxe générale PHP

Toute portion de code PHP doit être écrite dans une **balise spécifique : *<?php ...................... ?>*.**

On peut écrire un nombre quelconque de signes et lignes de code entre les marques de début (*<?php*) et de fin ( *?>*).

La syntaxe PHP est très proche de celle des langages modernes comme C#, Java ou JavaScript.

* PHP est sensible à la casse (majuscules/minuscules)
* PHP ignore les espaces, les tabulations et les sauts de lignes
* Les **commentaires** s'écrivent à l'aide des classiques*//* ou*/\* ... \*/*
* Le **point-virgule** termine une instruction
* les blocs sont définis entre **accolades** {...}

**Conseil**

**Une bonne pratique consiste à écrire une instruction par ligne en la terminant systématiquement par un point-virgule.**

**Un autre bonne pratique consiste à soigner l'indentation du code.**

Il restera toujours possible de *minifier* au final le code PHP pour alléger le volume des scripts à interpréter.

Les opérateurs arithmétiques et de comparaison restent très standards.

La **concaténation des chaînes de caractères est identifiée par l'opérateur .** (point).

Opérateurs logiques :

* ***and*** ou ***&&*** pour l'opérateur **ET**
* ***or*** ou***||*** pour l'opérateur **OU**
* ***!*** pour l'opérateur **NON**

Les variables PHP

**Pas de déclaration préalable** nécessaire pour les variables PHP ; le premier usage fait office de déclaration.

Les variables PHP sont identifiées par le **signe *$* en début de leur nom**.

Les variables PHP sont **typées dynamiquement** en fonction de leur contenu courant.

La portée d'une variable dépend de l'endroit où elle a été "déclarée"/initialisée : globale pour le script, locale à une fonction, voire locale à une structure de contrôle comme une boucle.

Si une fonction veut réutiliser une variable globale, celle-ci doit être **re-déclarée à l'intérieur de la fonction grâce au mot-clé** ***global***.

PHP propose des**types simples** comme les entiers ou caractères, un type **tableau de données** offrant quelques variantes bien pratiques, et un type **chaîne de caractères**. Des variables de type tableau ou chaînes sont automatiquement des objets dotées de propriétés et méthodes.

**Conseil**

**Le développeur doit soigner son usage des variables PHP ; une bonne pratique consiste à déclarer/initialiser chaque variable, à son bon niveau de portée, et d'effectuer des passations systématiques de paramètres aux fonctions**.

## Les variables supra-globales PHP

Le langage PHP étant destiné à la production des pages Web dynamiques, il est nécessaire de faire le lien avec le serveur Web et l'utilisateur.

PHP met à la disposition du développeur un certain nombre de **variables prédéfinies permettant de connaître et manipuler aisément le serveur Web, les données saisies en formulaire** par l'utilisateur ou encore de **suivre la navigation de page en page** des utilisateurs (notion de session).

Ces variables, appelées '**supra-globales**' sont reconnaissables par leur **nom en CAPITALES débutant par les caractères $\_**.

Les tableaux de variables en PHP

Les tableaux de variables PHP ne nécessitent **pas de déclaration ni d'instanciation** ; ils ne sont**pas typés** ; **leur taille s'adapte** au contenu courant ; l'**accès** à une cellule se fait classiquement **par indice** (initialisé à zéro).

Pour utiliser un tableau, il suffit d'affecter un contenu à une cellule quelconque.

Au-delà des tableaux classiques indicés, PHP inclut la notion de '**tableaux associatifs**' pour lesquels **l'accès à une cellule peut se faire aussi bien par un indice que par un libellé** choisi par le développeur.

Une syntaxe particulière à base de **l'opérateur *=>*** permet de définir le contenu d'un tableau associatif.

La plupart des variables système 'supra-globales' sont exposées sous forme de tableaux associatifs.

**Conseil**

Évitez de mixer les types de données dans un même tableau (au besoin, les objets PHP sont plus adaptés).

Le bon usage des tableaux et la maîtrise des tableaux associatifs sont les clés d'un développement efficace et de qualité en PHP.

## Les structures de contrôle en PHP

PHP propose des instructions classiques pour le contrôle du déroulement des programmes :

* structures conditionnelles ***if (...) {...} else {...} ;*** et **switch (...) case ... ; case... ;**
* structures répétitives **for (... ;... ;...) {...} ;** et **while (...) {...} ;**
* structures permettant de parcourir les éléments d'un tableau de variables (ou les membres d'un objet) : **foreach ($xxx as $yyy) {...}**

Procédures et fonctions en PHP

PHP propose seulement la structure de sous-programme ***function (...) {...} ;***

Si le sous-programme est terminé par une instruction ***return* suivie d'une valeur ou expression**, il se comporte somme une **fonction** retournant une valeur résultante ; sinon, le sous-programme se comporte comme une **procédure**.

Une ***function*** PHP peut recevoir des paramètres ; chaque paramètre est simplement doté d'un nom, local à la *function*, **sans type de donnée.**

**Conseil**

**Le développeur doit soigner la signature de ses fonctions et la compléter d'un commentaire précisant les types de données attendues et retournées.**

Gestion des erreurs en PHP

PHP supporte la classique **structure de gestion des erreurs d'exécution** *try {...} catch (...) {...}* et offre un **objet prédéfini *Exception*** qui expose le message d'erreur standard grâce à sa méthode*getMessage()*.

**Conseil**

**Après mise au point du code**, identifiez et **sécurisez systématiquement toute portion de code susceptible de lever des erreurs**, en particulier lors des accès aux bases de données ou à d'autres serveurs.

PHP et code HTML

Dans un script PHP, **tout code non inclus dans une balise *<?php ... ?>* est adressé en l'état au navigateur** de l'utilisateur.

PHP offre essentiellement **2 mots-clés pour générer du code HTML variable** à retourner à l'utilisateur :

* l'instruction ***echo ...*** ;
* la fonction ***print(...)*** ;

Leurs paramètres sont constitués de code source HTML (voire CSS ou JavaScript) écrit directement dans la commande ou résultat d'une expression à base de variables PHP.

Le code personnalisé contient en général une successions d'instructions constantes entrecoupées d'instructions variables.

Le développeur peut**entremêler le code constant et le code PHP** comme dans cet exemple :

***<?php****$Visites++;****?>***

*<h3>Le serveur renvoie un cookie de valeur : <br />*

***<?php****echo $Visites;****?>****. </h3>*

On parle de '**code spaghetti**'.

Le développeur peut aussi bien préparer dans une chaîne de caractères PHP tout un bloc de code avant de l'envoyer comme dans cet exemple :

***<?php****$Visites++;*

*$str= ‘ <h3>Le serveur renvoie un cookie de valeur : <br />' . $Visites . '</h3>';*

*echo $str;****?>****.*

**Conseil**

Quelle forme choisir ?

Le code spaghetti est semble-t-il moins efficace mais peut-être plus lisible que la forme plus algorithmique à base de concaténations.

Le seul conseil est de **rester cohérent** d'un bout à l'autre du script (et même du site Web).

Bufferisation du flux à envoyer au navigateur

Les commandes *echo* et *print()* alimentent directement le **flux de sortie qui est adressé au fur et à mesure au navigateur** client par le serveur Web.

Le développeur peut choisir de **préparer tout le code de la page Web avant son envoi** à l'aide des fonctions PHP *ob\_start()*, *ob\_get\_content()* et *ob\_end\_clean().*

**Conseil**

Cette technique de bufferisation est bien utile pour éviter un affichage 'par à-coups' de la page et dans le cas où une erreur serait détectée 'tard dans le code' PHP.

# PHP : Conseils d'expert (Ce qu'il faut savoir pour bien utiliser le langage PHP pour des développements

## PHP et les chaînes de caractères

Le langage PHP est très complet pour gérer les chaînes de caractères.

De nombreuses **fonctions** du langage comme strlen(), trim() ou substr() permettent de **manipuler** les chaînes.

D'autres **fonctions** comme printf() ou sprintf() permettent de **formater** un texte.

PHP inclut des **fonctions particulières pour traiter du code HTML** :

* htmlspecialchars() : convertit les caractères spéciaux comme les lettres accentuées ou les signes < et > en ‘entités' HTML (&eacute; &gt;...)
* htmlentities() : même principe mais étendu à plus de cas de figure
* html\_entity\_decode() : convertit à l'inverse les entités HTML en caractères ordinaires

Redirection vers une autre page

La fonction PHP***header("location :...")*** permet de **rediriger l'utilisateur vers une autre page que celle demandée** (cas d'erreur, session expirée...).

La fonction *header()* permet en outre de **modifier les paramètres de l'entête HTTP du message retourné** au navigateur client ; au-delà de la redirection, cette fonction est parfois bien utile pour réaliser un rafraîchissement automatique de la page (paramètre *refresh :...*) ou pour jouer finement sur le protocole d'échanges entre client, routeurs et serveur Web (paramètres *Content-Type :...*, *Cache-control :...* ou *Expires :...*)

**Conseil**

Pour gérer les accès sécurisés aux pages d'un site, **testez systématiquement l'existence et le contenu des variables de session dédiées et redirigez au besoin l'utilisateur vers la page de login** grâce à une fonction*header()*.

## PHP et les formulaires HTML

Le langage PHP est idéal pour récupérer les données, saisies par l'utilisateur en formulaire HTML, afin de moduler les traitements ou de stocker des informations en bases de données.

Un formulaire HTML peut passer les données au serveur Web directement dans l'URL de la page demandée (method='get') ou dans le corps du message HTTP (method='post') ; PHP expose ces données à travers les 2 **tableaux associatifs supra-globaux** **$\_GET[]** et **$\_POST[]** ; et le tableau**$\_REQUEST[]** regroupe tous les paramètres passés en get, en post et même par cookies.

**Seuls les contrôles graphiques HTML nommés** (attribut HTML 'name') sont adressés au serveur.

# PHP : Conseils d'expert (Ce qu'il faut savoir pour bien utiliser le langage PHP pour des développements

## Cas des boutons-radio

Au contraire de JavaScript qui voit tous les boutons placés sur la page, **PHP voit une seule variable ayant la valeur du bouton choisi** par l'utilisateur.

Cas des cases à cocher

**Seules les cases cochées par l'utilisateur sont transmises du navigateur au serveur Web**.

**Conseil**

* PHP voit les cases à cocher comme des**variables différentes** si elles portent des **noms HTML différents** (ou ne les voit pas si elles n'ont pas été cochées ; tester l'existence par *if(isset($...)*).
* PHP voit les cases à cocher comme un **tableau de variables** (éventuellement inexistant si rien n'a été coché) si elles portent le **même nom HTML** ;

Tester d'abord qu'au moins une case a été cochée : *if(isset($\_GET["idMessage"]))*

Tester éventuellement le nombre de données reçues : *count($\_GET["idMessage"])*

Visiter toutes les valeurs reçues avec une boucle :*foreach ($\_GET["idMessage"] as $unIdMessage) {...}*

PHP et les sessions

Le protocole HTTP gère les échanges entre client et serveur Web. HTTP est un protocole sans conservation d'état ("sitôt servi, sitôt oublié").

Les sessions permettent d'**enregistrer des variables**, de les préserver pour les **réutiliser au fil des pages visitées** par un même utilisateur manipulant un même navigateur.

Une session ‘tombe' d'elle-même après un certain temps d'inactivité.

**Conseil**

Attention que ce mécanisme est gourmand en ressources serveur : il n'est pas activé par défaut en PHP et **le développeur doit activer le mécanisme sur chaque page nécessitant un accès aux variables de session**.

Le tableau prédéfini supra-global***$\_SESSION[]*** **permet d'accéder très facilement aux variables de session de l'utilisateur courant**.

Le développeur **active** le mécanisme des sessions à l'aide de la fonction ***session\_start()***.

Il **teste l'existence** d'une variable de session à l'aide de la fonction ***isset($\_SESSION["xxx"])***.

Il **détruit** intentionnellement toutes les variables de session de l'utilisateur courant à l'aide de la fonction ***session\_destroy()***.

**Conseil**

Pour gérer la connexion/déconnexion d'un utilisateur, utiliser les variables de session.

PHP et les bases

// Les types de variables

// Entiers (integer)

$nombre = 85 ;

//Décimaux (float)

$nombre2 = 82.2 ;

// Chaîne de caractères (string)

$chaine = « ceci est ub texte » ;

// Booléen (boolean)

$booleen = true ; //ou false

// Connaître le contenu et le type d’une variable

Var\_dump($nombre) ;