Ressourcesinformatiques

+ QUIZ

Version en ligne
OFFERTE !
pendant 1 an

PHP 8

Développez un site web dynamique et interactif

En téléchargement

- code des exemples
- solutions des exercices

Olivier HEURTEL

eni

Les éléments à télécharger sont disponibles à l'adresse suivante :
http://www.editions-eni.fr
Saisissez la référence ENI de l'ouvrage **RI8PHP** dans la zone de recherche et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

Chapitre 1
Préambule

1. Objectif de l'ouvrage 11
2. Bref historique de PHP 12
3. Où se procurer PHP ? 13
4. Conventions d'écriture 14
5. À propos des exercices 15

Chapitre 2
Introduction à PHP

1. Qu'est-ce que PHP ? 17
2. Structure de base d'une page PHP 19
   2.1 Les balises PHP 19
   2.2 La fonction echo 19
   2.3 Séparateur d'instructions 21
   2.4 Commentaires 22
   2.5 Mixer du PHP et de l'HTML 22
   2.6 Règles de nommage 26
   2.7 Exercice 1 : mon premier script PHP 26
3. Configuration de PHP 27
   3.1 Le fichier de configuration php.ini 27
   3.2 Informations sur la configuration 28
   3.3 Jeu de caractères 31
4. Utiliser PHP en ligne de commande 31
5. Les bases du langage PHP 32
   5.1 Constantes 32
      5.1.1 Définition 32

5.1.2 Portée 35
5.2 Variables 35
5.2.1 Initialisation et affectation 35
5.2.2 Portée et durée de vie 37
5.2.3 Variables dynamiques (ou variables variables) 38
5.3 Types de données 38
5.3.1 Types de données disponibles 38
5.3.2 Types de données scalaires 39
5.3.3 Types de données spéciaux 47
5.3.4 Déclaration de type 47
5.4 Tableaux 49
5.4.1 Définition 49
5.4.2 Création 50
5.4.3 Manipulation 56
5.4.4 Décomposer un tableau 61
5.4.5 Portée 63
5.5 Opérateurs 63
5.5.1 L'opérateur d'affectation par valeur 63
5.5.2 L'opérateur d'affectation par référence 65
5.5.3 Les opérateurs arithmétiques 66
5.5.4 L'opérateur de chaîne 67
5.5.5 Les opérateurs combinés 67
5.5.6 Les opérateurs de comparaison 68
5.5.7 Les opérateurs logiques 69
5.5.8 L'opérateur ternaire 70
5.5.9 L'opérateur de fusion NULL 71
5.5.10 L'opérateur d'affectation de fusion NULL 72
5.5.11 L'opérateur de comparaison combinée 73
5.5.12 Précédence des opérateurs 74
5.6 Structures de contrôle 75
5.6.1 La structure if 75
5.6.2 La structure switch 77
5.6.3 La structure while 80
5.6.4 La structure do ... while 82
5.6.5 La structure for 83
5.6.6 Les instructions continue et break 86
5.6.7 L'expression match 87

5.7 Inclure un fichier. 90
5.7.1 Fonctionnement 90
5.7.2 Utilisation 92
5.8 Interrompre le script. 93
5.9 Exercice 2 : variables et structures de contrôle 94

Chapitre 3
Utiliser les fonctions PHP

1. Préambule. 101
2. Manipuler les constantes, les variables et les types de données 102
2.1 Constantes 102
2.2 Variables 103
2.3 Types de données 109
2.3.1 Conversions 109
2.3.2 Fonctions utiles 114
3. Manipuler les tableaux 120
4. Manipuler les nombres 134
5. Manipuler les chaînes de caractères 139
6. Utiliser les expressions rationnelles 158
6.1 Introduction 158
6.2 Structure d'une expression rationnelle 158
6.3 Fonctions. 168
7. Manipuler les dates 173
8. Générer un identifiant unique 189
9. Manipuler les fichiers sur le serveur. 191
9.1 Fonctions utiles. 191
9.2 Exemples d'utilisation. 199
10. Manipuler les en-têtes HTTP 200
11. Exercices. 202
11.1 Exercice 3 : manipuler les données 202
11.2 Exercice 4 : écrire et lire un fichier sur le serveur 207

Chapitre 4
Écrire des fonctions et des classes PHP

1. Fonctions 211
   1.1 Introduction 211
   1.2 Déclaration et appel 211
   1.3 Paramètres 221
      1.3.1 Syntaxe 221
      1.3.2 Valeur par défaut 222
      1.3.3 Déclaration du type de données 224
      1.3.4 Passage par référence 227
      1.3.5 Liste variable de paramètres 229
      1.3.6 Utilisation du nom du paramètre dans l'appel 231
   1.4 Considérations sur les variables utilisées dans les fonctions 233
      1.4.1 Variables locales/globales 233
      1.4.2 Variables statiques 235
   1.5 Les constantes et les fonctions 236
   1.6 Récursivité 237
   1.7 Fonction anonyme 239
   1.8 Fonction fléchée 240
   1.9 Fonction générateur 242
   1.10 Exercice 5 : écrire des fonctions 245
2. Classes 246
   2.1 Concept 246
   2.2 Définir une classe 247
   2.3 Instancier une classe 252
   2.4 Héritage 255
   2.5 Autres fonctionnalités sur les classes 259
      2.5.1 Classes ou méthodes abstraites 259
      2.5.2 Classes ou méthodes finales 260
      2.5.3 Interfaces 261
      2.5.4 Propriétés ou méthodes statiques - Constantes de classes . . . 263
      2.5.5 Traits 266
      2.5.6 Classes anonymes 269
   2.6 Exceptions 270
   2.7 Exercice 6 : écrire une classe 274
3. Espaces de noms 277

Chapitre 5
Gérer les erreurs dans un script PHP

1. Vue d'ensemble 283
2. Les messages d'erreur PHP. 284
3. Les fonctions de gestion des erreurs 288
4. Exercice 7 : gérer les erreurs 303

Chapitre 6
Gérer les formulaires et les liens

1. Vue d'ensemble 307
   1.1 Introduction 307
   1.2 Les liens 307
   1.3 Les formulaires 310
      1.3.1 Petit rappel sur les formulaires 310
      1.3.2 Construire un formulaire dynamiquement 312
      1.3.3 Traiter un formulaire à l'aide d'un script PHP. 317
   1.4 Récupérer les données d'une URL ou d'un formulaire 321
2. Récupérer les données passées par l'URL. 324
   2.1 Considérations 324
      2.1.1 Que se passe-t-il si deux paramètres portent le même nom ? 324
      2.1.2 Utiliser un tableau pour passer des données dans l'URL. 324
   2.2 Transmettre des caractères spéciaux. 325
   2.3 Exercice 8 : récupérer des données passées par l'URL. 328
3. Récupérer les données saisies dans le formulaire. 330
   3.1 Considérations 330
      3.1.1 Que se passe-t-il si deux zones portent le même nom ? 330
      3.1.2 Que se passe-t-il s'il y a deux formulaires dans la page HTML ?. 331
      3.1.3 Utiliser un tableau pour récupérer les données saisies 331
      3.1.4 Passer des informations dans une zone de formulaire cachée. 332
   3.2 Les différents types de zones 334
      3.2.1 Vue d'ensemble 334
      3.2.2 Zones contenant du texte. 337
      3.2.3 Groupes de boutons radio 337

3.2.4 Cases à cocher 338
3.2.5 Listes à sélection unique 340
3.2.6 Listes à sélection multiple 341
3.2.7 Boutons de validation 343
3.2.8 Boutons image 345
3.2.9 Boutons "reset" ou "button" 346
3.3 Synthèse 346
3.4 Exercice 9 : récupérer des données saisies dans un formulaire 348
4. Contrôler les données récupérées 351
4.1 Vue d'ensemble 351
4.2 Vérifications classiques 352
4.2.1 Nettoyage des espaces indésirables 352
4.2.2 Données obligatoires 352
4.2.3 Longueur maximum d'une chaîne 352
4.2.4 Caractères autorisés pour une chaîne - Format 353
4.2.5 Validité d'une date - Plage de valeurs 353
4.2.6 Validité d'un nombre - Plage de valeurs 355
4.2.7 Validité d'une adresse e-mail 356
5. Problèmes sur les données récupérées 357
6. Utilisation des filtres 364
6.1 Principes 364
6.2 Application aux formulaires 373
6.3 Exercices 375
6.3.1 Exercice 10 : contrôler des données passées par l'URL 375
6.3.2 Exercice 11 : contrôler des données saisies dans un formulaire 376
7. Aller sur une autre page 380
8. Échanger un fichier entre le client et le serveur 386
8.1 Vue d'ensemble 386
8.2 Envoyer un fichier depuis le client (upload) 387
8.3 Télécharger un fichier à partir du serveur (download) 392

Chapitre 7
Accéder aux bases de données

1. Introduction 399
   - 1.1 Vue d'ensemble 399
   - 1.2 La notion de fetch 400
2. Utilisation de MySQL 401
   - 2.1 Préambule 401
   - 2.2 Connexion et déconnexion 402
     - 2.2.1 Connexion 402
     - 2.2.2 Déconnexion 403
     - 2.2.3 Obtenir des informations sur le serveur MySQL 404
     - 2.2.4 Obtenir des informations en cas d'erreur de connexion 404
     - 2.2.5 Exemple 404
   - 2.3 Sélectionner une base de données 406
   - 2.4 Utiliser des requêtes non préparées 407
     - 2.4.1 Vue d'ensemble 407
     - 2.4.2 Exécuter une requête 407
     - 2.4.3 Connaître le nombre de lignes dans le résultat d'une requête de lecture409
     - 2.4.4 Extraire le résultat d'une requête de lecture 410
     - 2.4.5 Obtenir des informations sur le résultat d'une requête de mise à jour 418
     - 2.4.6 Gérer les erreurs 421
   - 2.5 Utiliser des requêtes préparées 422
     - 2.5.1 Vue d'ensemble 422
     - 2.5.2 Préparer une requête 424
     - 2.5.3 Lier des variables PHP aux paramètres de la requête 425
     - 2.5.4 Exécuter la requête préparée 427
     - 2.5.5 Lier des variables PHP aux colonnes du résultat d'une requête de lecture 428
     - 2.5.6 Extraire le résultat d'une requête de lecture 430
     - 2.5.7 Utiliser un résultat stocké 431
     - 2.5.8 Obtenir des informations sur le résultat d'une requête de mise à jour 434
     - 2.5.9 Gérer les erreurs 437
     - 2.5.10 Fermer une requête préparée 438
   - 2.6 Gérer les transactions 438

2.7 Appeler un programme stocké 441
2.7.1 Procédure stockée 441
2.7.2 Fonction stockée 445
2.8 Exercice 12 : utiliser MySQL 447
3. Utilisation d'Oracle 455
3.1 Préambule 455
3.2 Environnement NLS 455
3.3 Connexion et déconnexion 456
3.3.1 Connexion 456
3.3.2 Déconnexion 458
3.3.3 Obtenir des informations sur le serveur Oracle 459
3.3.4 Obtenir des informations en cas d'erreur de connexion 459
3.3.5 Exemple 459
3.4 Exécuter une requête 461
3.4.1 Vue d'ensemble 461
3.4.2 Analyser une requête 462
3.4.3 Lier des variables PHP aux paramètres de la requête 463
3.4.4 Exécuter une requête 466
3.4.5 Extraire le résultat de la requête de lecture 468
3.4.6 Mettre à jour des données et gérer les transactions 481
3.4.7 Fermer un curseur 487
3.5 Appeler une procédure stockée 487
3.6 Illustration des problèmes liés à l'environnement NLS 493
3.7 Gérer les erreurs 495
3.8 Exercice 13 : utiliser Oracle 498
4. PHP Data Objects (PDO) 505
5. Gestion des apostrophes dans le texte des requêtes 508
6. Exemples d'intégration dans des formulaires 513
6.1 Vue d'ensemble 513
6.2 Construction d'une liste de sélection dans un formulaire 524
6.3 Affichage d'une liste 526
6.4 Formulaire de saisie en liste 530
6.5 Formulaire de recherche et de saisie 534

Chapitre 8
Gérer les sessions

1. Description du problème 539
2. Authentification 540
   2.1 Vue d'ensemble 540
   2.2 Saisie de l'identification 540
      2.2.1 Identification par formulaire 540
      2.2.2 Identification par authentification HTTP 543
   2.3 Vérifier l'identification saisie 545
3. Utiliser des cookies 546
   3.1 Principe 546
   3.2 Application à la gestion des sessions 553
4. Utiliser la gestion des sessions de PHP 554
   4.1 Principes 554
   4.2 Mise en œuvre 554
   4.3 Gérer soi-même la transmission de l'identifiant de session 568
      4.3.1 Description du problème 568
      4.3.2 Solution 572
   4.4 Quelques directives de configuration supplémentaires 574
   4.5 Exemples d'application 575
      4.5.1 Principes 575
      4.5.2 Avec authentification des utilisateurs 578
   4.6 Remarques et conclusion 581
   4.7 Exercice 14 : gérer les sessions 585
5. Conserver des informations d'une visite à une autre 589
6. Petite synthèse sur les variables Get/Post/Cookie/Session 595

Chapitre 9
Envoyer un courrier électronique

1. Vue d'ensemble 599
2. Envoyer un message texte sans pièce jointe 599
3. Envoyer un message au format MIME 603
   3.1 Préambule 603
   3.2 Message au format HTML 603
   3.3 Message avec pièce jointe 606
4. Exercice 15 : envoyer un courrier électronique 609

Annexe

1. Variables PHP prédéfinies 613
2. Constantes PHP prédéfinies 615
3. Exemples complémentaires 616
   3.1 Introduction 616
   3.2 Lire un document XML 617
   3.3 Générer un document PDF 622
   3.4 Générer une image 626
4. Résumé des principales nouveautés de la version 8 631

Index 635

Chapitre 4
Écrire des fonctions et des classes PHP

1. Fonctions

1.1 Introduction

À l'instar des différents langages de développement, PHP offre la possibilité de définir ses propres fonctions (appelées fonctions "utilisateur") avec tous les avantages associés (modularité, capitalisation...). Une fonction est un ensemble d'instructions identifiées par un nom, dont l'exécution retourne une valeur et dont l'appel peut être utilisé comme opérande dans une expression. Une procédure est un ensemble d'instructions identifiées par un nom qui peut être appelé comme une instruction.

1.2 Déclaration et appel

Le mot-clé `function` permet d'introduire la définition d'une fonction.

Syntaxe

```
function nom_fonction([paramètre]) [: type]{
  instructions;
}
```

`nom_fonction` Nom de la fonction (doit respecter les règles de nommage présentées dans le chapitre Introduction à PHP - Structure de base d'une page PHP). Ce nom n'est pas sensible à la casse (pour PHP, les fonctions `unefonction` et `UneFonction` sont les mêmes).

`paramètre` Paramètres éventuels de la fonction exprimés sous forme d'une liste de variables (cf. section Fonctions - Paramètres) : `$paramètre1, $paramètre2, ...`

`type` Déclaration du type de données retourné par la fonction. Valeurs possibles : `int`, `float`, `string`, `bool`, `array`, `callable`, `iterable`, `object`, `mixed`, `void`, un nom de classe ou d'interface (cf. dans ce chapitre la section Classes), ou une union de types. Le nom du type peut être précédé d'un point d'interrogation (`?`) qui indique que la fonction peut retourner une valeur `NULL`. Voir le chapitre Les bases du langage PHP pour la définition des types de données (section Les bases du langage PHP - Types de données).

`instructions` Ensemble des instructions qui composent la fonction.

Le nom de la fonction ne doit pas être un mot réservé PHP (nom de fonction native, d'instruction) ni être égal au nom d'une autre fonction préalablement définie.

Une fonction utilisateur peut être appelée comme une fonction native de PHP : dans une affectation, dans une comparaison, etc.

Si la fonction retourne une valeur, il est possible d'utiliser l'instruction `return` pour définir la valeur de retour de la fonction.

Syntaxe

```
return expression;
```

`expression` Expression dont le résultat constitue la valeur de retour de la fonction (`NULL` par défaut).

Le résultat d'une fonction peut être de n'importe quel type (chaîne, nombre, tableau, etc.).

L'instruction `return` stoppe l'exécution de la fonction et retourne le résultat de `expression` à l'appelant. Si plusieurs instructions `return` sont présentes dans la fonction, c'est la première rencontrée dans le déroulement des instructions qui définit la valeur de retour et provoque l'interruption de la fonction. Si la fonction ne comporte aucune instruction `return` (ou si aucune instruction `return` n'est exécutée), la valeur de retour de la fonction est `NULL`.

© Editions ENI - All rights reserved

Exemple

```
<?php
// Fonction sans paramètre qui affiche "Bonjour !"
// Pas de valeur de retour.
function afficher_bonjour() {
  echo 'Bonjour !<br />';
}
// Fonction avec deux paramètres qui retourne le produit
// des deux paramètres.
function produit($valeur1,$valeur2) {
  return $valeur1 * $valeur2;
}
// Appel de la fonction afficher_bonjour.
afficher_bonjour();
// Utilisations de la fonction produit :
// - dans une affectation
$résultat = produit(2,4);
echo "2 x 4 = $résultat<br />";
// - dans une comparaison
if (produit(10,12) > 100) {
  echo '10 x 12 est supérieur à 100.<br />';
}
?>
```

Résultat

```
Bonjour !
2 x 4 = 8
10 x 12 est supérieur à 100.
```

Remarque

Dans le langage PHP, il n'existe pas à proprement parler de procédure. Pour définir quelque chose d'équivalent à une procédure, il suffit de définir une fonction qui ne retourne pas de valeur et d'appeler la fonction comme s'il s'agissait d'une instruction (comme la fonction `afficher_bonjour` par exemple). Une fonction qui ne retourne rien peut explicitement être déclarée avec le type de retour `void`.

Comme nous l'avons déjà évoqué, le contenu d'un tableau peut être transformé en liste de paramètres dans un appel de fonction grâce à l'opérateur `...` (points de suspension).

Exemple

```
<?php
// Fonction avec trois paramètres qui retourne la somme
// des trois paramètres.
function somme($valeur1,$valeur2,$valeur3) {
  return $valeur1 + $valeur2 + $valeur3;
}
// Transformation du contenu d'un tableau en
// liste de paramètres.
$valeurs = [1,2,3];
echo '1 + 2 + 3 = ',somme(...$valeurs),'<br />';
// La même chose pour une partie seulement des paramètres
// avec un tableau défini directement dans l'appel.
echo '1 + 2 + 4 = ',somme(1,...[2,4]),'<br />';
?>
```

Résultat

```
1 + 2 + 3 = 6
1 + 2 + 4 = 7
```

Lorsqu'une fonction retourne un tableau, il est possible d'accéder directement à un élément du tableau lors de l'appel à la fonction avec une syntaxe du type `fonction(...)[clé]`.

Exemple

```
<?php
// Définition d'une fonction qui retourne un tableau.
function qui() {
  return ['Olivier','Heurtel'];
}
// Appel de la fonction et récupération directe du prénom stocké
// à l'indice 0 du tableau retourné.
$prénom = qui()[0];
echo "qui()[0] = $prénom<br />";
?>
```

Résultat

```
qui()[0] = Olivier
```

Cette technique fonctionne aussi lorsque la fonction retourne un tableau multidimensionnel avec une syntaxe du type `fonction(...)[clé1][clé2]`.

Il est possible d'utiliser une fonction avant de la définir.

© Editions ENI - All rights reserved

Exemple

```
<?php
// Utilisation de la fonction produit.
echo produit(5,5);
// Définition de la fonction produit.
function produit($valeur1,$valeur2) {
  return $valeur1 * $valeur2;
}
?>
```

Résultat

```
25
```

Il n'y a donc aucun problème pour définir des fonctions qui s'appellent entre elles.

Remarque

Une fonction est utilisable uniquement dans le script où elle est définie. Pour l'employer dans plusieurs scripts, il faut, soit recopier sa définition dans les différents scripts (vous perdez l'intérêt de définir une fonction), soit la définir dans un fichier inclus partout où la fonction est nécessaire.

Exemple

– Fichier `fonctions.inc` contenant des définitions de fonctions :

```
<?php
// Définition de la fonction produit.
function produit($valeur1,$valeur2) {
  return $valeur1 * $valeur2;
}
?>
```

– Script utilisant les fonctions définies dans `fonctions.inc` :

```
<?php
// Inclusion du fichier contenant la définition des fonctions.
include('fonctions.inc');
// Utilisation de la fonction produit.
echo produit(5,5);
?>
```

Déclaration du type de retour

Il est possible de définir le type de données retourné par une fonction.

Lorsque c'est le cas, dans le mode de fonctionnement par défaut (à opposer au mode strict présenté ci-dessous), PHP effectue si besoin une conversion automatique de la valeur retournée dans le type de données déclaré.

Exemple

```
<?php
// Déclaration de deux fonctions qui retournent le produit
// des deux paramètres, la deuxième spécifiant un type
// de données "entier" pour la valeur de retour.
function produit1($valeur1,$valeur2) {
  return $valeur1 * $valeur2;
}
function produit2($valeur1,$valeur2) : int {
  return $valeur1 * $valeur2;
}
// Appel des deux fonctions avec les mêmes paramètres
echo 'produit1(20,1/7) => ',var_dump(produit1(20,1/7)),'<br />';
echo 'produit2(20,1/7) => <b>',var_dump(produit2(20,1/7)),'</b><br />';
?>
```

Résultat

```
produit1(20,1/7) => float(2.8571428571428568)
produit2(20,1/7) => int(2)
```

Sur cet exemple, nous voyons bien que la valeur retournée par la deuxième fonction a été convertie en entier par PHP (avec les règles de conversion évoquées dans le chapitre Introduction à PHP - Les bases du langage PHP - Types de données).

Si PHP n'est pas en mesure d'effectuer la conversion (types de données non convertibles entre eux), une exception `TypeError` est levée ; cette exception interrompt le script si elle n'est pas gérée (cf. dans ce chapitre la section Classes - Exceptions).

Exemple

```
<?php
// Déclaration et appel d'une fonction qui doit retourner un
// tableau mais qui retourne une chaîne de caractères.
function qui() : array {
  return 'Olivier Heurtel';
}
echo 'qui()[0] = ',qui()[0];
?>
```

Résultat

```
qui()[0] =
Fatal error: Uncaught TypeError: Return value of qui() must be of the
type array, string returned in /app/scripts/index.php:5 Stack trace: #0
/ app/scripts/index.php(7): qui() #1 {main} thrown in/app/scripts/
index.php on
line 5
```

© Editions ENI - All rights reserved

Une fonction déclarée avec un type de retour autre que `void` doit retourner une valeur non `NULL`. Si ce n'est pas le cas, une erreur est retournée, différente selon les cas :

Absence d'instruction `return`

```
Fatal error: Uncaught TypeError: Return value of MaFonction() must be of
the type int, none returned in ...
```

Instruction `return` vide

```
Fatal error: A function with return type must return a value in ...
```

Instruction `return NULL`

```
Fatal error: Uncaught TypeError: Return value of MaFonction() must be of
type int, null returned in ...
```

Pour autoriser une fonction à retourner une valeur `NULL`, il faut faire précéder le nom du type (autre que `void`) d'un point d'interrogation (`?`).

```
<?php
// Déclaration et appel d'une fonction qui spécifie un
// type de donnée de retour qui peut être NULL
function cube($valeur) : ?int {
  if (is_null($valeur)) {
    return NULL;
  } else {
    return $valeur ** 3 ;
  }
}
echo 'cube(2) => <b>',var_dump(cube(2)),'</b><br />';
echo 'cube(NULL) => <b>',var_dump(cube(NULL)),'</b><br />';
?>
```

Résultat

```
cube(2) => int(8)
cube(NULL) => NULL
```

Même avec cette option, la fonction doit avoir une instruction `return` non vide. Si ce n'est pas le cas, une erreur est retournée, différente selon les cas :

Absence d'instruction `return`

```
Fatal error: Uncaught TypeError: Return value of MaFonction() must be of
type ?int, none returned in ...
```

Instruction `return` vide

```
Fatal error: A function with return type must return a value (did you mean
"return null;" instead of "return;"?) in ...
```

À l'inverse, il est possible de déclarer qu'une fonction ne retourne rien, en utilisant `void` comme nom de type. Dans ce cas, la fonction doit omettre l'instruction `return` ou mettre une instruction `return` vide, sans valeur (même pas `NULL`).