



Les fonctions et la récupération de données

Technologies web



Les fonctions

Les fonctions et la récupération de données

Les fonctions en quelques mots... (1/2)

- Comme la plupart des langages de programmation, **PHP** propose à ses utilisateurs de créer leurs propres **fonctions** en complément de celles proposées nativement par le langage
- Écrire des **fonctions** permet d'éviter la production de code **dupliqué** et **redondant** mais aussi de répondre à un **problème complexe** en le découpant en plusieurs **petits problèmes**
- Concrètement, une **fonction** est définie par une **suite d'instructions** - Elle peut être perçue comme un **sous-programme** au **programme principal** l'appelant

Les fonctions en quelques mots... (2/2)

- Les **fonctions** peuvent être séparées en **2 types** :
 - Les **procédures** qui **ne renvoient pas de valeur** à la fin de leur exécution
 - Les **fonctions** qui **renvoient forcément une valeur** de type **primitif** (`int`, `float`, `string`, `boolean`) ou **structuré** (`array`, `object`)
- Une **fonction** peut également prendre (ou non) des **paramètres d'entrées** de type **primitif** ou **structuré**
- Une **fonction** est caractérisée par **2 éléments principaux** :
 - Sa **signature** : composée du **nom de la fonction** et de la **liste de ses paramètres**
 - Son **corps** : contient la **suite d'instructions** à exécuter

Déclarer une fonction PHP

- La **déclaration** d'une nouvelle **fonction PHP** passe par le mot-clé `function` suivi du **nom** que vous souhaitez lui donner et de ses **paramètres** éventuels

```
10  <?php
11      function somme($a, $b) {
12          $somme = $a + $b;
13          echo "La somme de {$a} et {$b} vaut $somme";
14      }
15  ?>
```

- Notez que les noms de la **fonction** et de ses **paramètres** sont libres (en respectant le **camel-case**)
- Une **fonction** ne doit être déclarée qu'**une seule fois**

Déclarer une fonction PHP - Valeur par défaut

- Il est possible de donner une **valeur par défaut** à un **paramètre** d'une **fonction** - Ce dernier prendra cette valeur si elle n'a pas été définie lors de son **appel**

```
10  <?php
11      function somme($a, $b = 5) {
12          $somme = $a + $b;
13          echo "La somme de {$a} et {$b} vaut $somme";
14      }
15  ?>
```

Déclarer une fonction PHP - Valeur de retour (1/2)

- Jusqu'ici, tous nos exemples étaient des **procédures** étant donné qu'elles ne renvoient **aucune valeur de retour**
- Modifions notre **fonction** pour qu'elle **renvoie** la somme de nos 2 **paramètres**

```
10  <?php
11      function somme($a, $b) {
12          $somme = $a + $b;
13          // Renvoie la somme au programme principal
14          return $somme;
15      }
16  ?>
```


Déclarer une fonction PHP - Valeur de retour (2/2)

- Le mot-clé `return` est utilisé pour **renvoyer une valeur** - Notez qu'il **stoppe** également l'exécution de la **fonction**
- Par ailleurs, une même **fonction** peut comporter plusieurs instructions `return`

```
10  <?php
11      function estConnecte($connecte) {
12          // Test du paramètre d'entrée
13          if($connecte === 1) {
14              return true;
15          }
16          return false;
17      }
18  ?>
```


Déclarer une fonction PHP - Retour de plusieurs valeurs

- PHP autorise le **renvoi d'un tableau** pour permettre aux **fonctions** de renvoyer **plusieurs valeurs**

```
10  <?php
11      function identifiantsSql() {
12          return ["localhost", "root", "root", "maBaseDeDonnees"];
13      }
14  ?>
```

Déclarer une fonction PHP - Typage

- Même si le **typage** n'est pas obligatoire en **PHP**, il est possible de définir un type aux paramètres d'une fonction et à sa valeur de retour attendue
- Cette pratique offre une meilleure lisibilité et aide au débogage

```
10  <?php
11      // Déclaration de la fonction 'somme()'
12      function somme(int $a, int $b): int {
13          return $a + $b;
14      }
15  ?>
```

Appeler une fonction PHP

- Nous savons **déclarer** une **fonction** mais pas encore l'utiliser - Pour cela, il nous faut **l'appeler**
- **L'appel** d'une **fonction** se fait généralement depuis le **programme principal** après l'avoir **déclaré**
- Une **fonction** est appelée grâce à son **nom** suivi de ses **arguments** éventuels

Appeler une fonction PHP

```
10  <?php
11      // Déclaration de la fonction 'somme()'
12      function somme($a, $b) {
13          $somme = $a + $b;
14          echo "La somme de {$a} et {$b} vaut {$somme}<br>";
15      }
16
17      // 3 appels à la fonction 'somme()'
18      somme(1, 5);
19      somme(7, 3);
20      somme(8, 4);
21  ?>
```

La somme de 1 et 5 vaut 6
La somme de 7 et 3 vaut 10
La somme de 8 et 4 vaut 12

Appeler une fonction PHP

- L'**appel** à la **fonction** renvoyant la somme en **retour** grâce au **mot-clé** `return`, n'affichera rien car elle ne propose aucun `echo`
- Il nous faudra donc l'afficher dans le **programme principal** en prenant éventuellement soin de **stocker le résultat** (la valeur renvoyée par la **fonction**) dans une **structure de données**

Appeler une fonction PHP

```
10  <?php
11      // Déclaration de la fonction 'somme()'
12      function somme($a, $b) {
13          return $a + $b;
14      }
15
16      $sommes = array();
17      // Ajoute les retours de la fonction 'somme()' au tableau 'sommes'
18      $sommes[] = somme(1, 5);
19      $sommes[] = somme(7, 3);
20      $sommes[] = somme(8, 4);
21
22      echo "<pre>";
23      print_r($sommes);
24      echo "</pre>";
25  ?>
```

```
Array
(
    [0] => 6
    [1] => 10
    [2] => 12
)
```



Les importations de fichiers

Les fonctions et la récupération de données

Importer des fichiers

- Dans les sites web dynamiques, il peut être intéressant de **réutiliser** des parties de code **PHP** déclarées dans des **fichiers différents**
- Certaines **fonctions PHP** proposent justement d'**importer** et exécuter du code **PHP** réutilisable depuis un **fichier différent**
- Ce sont les **fonctions** `include()` et `require()`

Les fonctions `include()` et `require()`

- `include()` renvoie une **erreur** de type `WARNING` si elle ne parvient pas à ouvrir le fichier passé en **argument** - Ainsi le code qui suit sera **exécuté** malgré tout
- `require()` lance une **erreur** de type `FATAL` interrompant l'exécution du code
- `require()` sera donc préférable dans le cas où le **fichier importé** doit être absolument **présent** dans le reste du programme
- Ces 2 **fonctions** prennent **un seul paramètre** sous forme de **chaîne de caractères** représentant le **chemin menant au fichier à importer**

Les fonctions `include()` et `require()`

```
10  <?php
11      // Importations avec la fonction 'require()'
12      require("./fichier.php");
13      require '../dossier/fichier.php';
14
15      // Importations avec la fonction 'include()'
16      include("./fichier.php");
17      include '../dossier/fichier.php';
18  ?>
```

- Comme pour la fonction `echo()`, les **parenthèses** sont **optionnelles**
- En important un fichier **PHP**, son code est exécuté et les **variables**, **objets**, **tableaux**, etc. peuvent être utilisés dans la suite du programme

Les fonction `include()` et `require()`

```
10  <?php
11      # Dans le fichier config.php
12      // Définition des variables '$a' et '$b'
13      $a = 15;
14      $b = 5;
15  ?>
```

```
10  <?php
11      # Dans un fichier différent
12      // Importation et exécution du fichier 'config.php'
13      require("./config.php");
14      // Calcul de la somme de '$a' et '$b'
15      $somme = $a + $b;
16      // Affichage de la somme
17      echo "Somme de $a + $b = $somme";
18  ?>
```

La récupération de données de formulaire

Les fonctions et la récupération de données

Les tableaux superglobaux `$_GET` et `$_POST`

- Les **variables superglobales** `$_GET` et `$_POST` représentent chacune un **tableau associatif**
- Comme toutes les **variables superglobales**, elles sont générées par **PHP** avant même que la première ligne du script ne soit exécutée et sont visibles et accessibles en **lecture** et en **écriture** partout dans le programme
- La **tableau** `$_GET` contient tous les couples **variables / valeurs** transmis à travers l'**URL**
- Le tableau `$_POST` contient tous les couples **variables / valeurs** transmis à travers une **requête HTTP POST**

Les tableaux superglobaux `$_GET` et `$_POST`

```
10  <?php
11  |      echo $_GET["prenom"];
12  ?>
13
14  <form method="get">
15  |      <input type="text" name="prenom">
16  |      <input type="submit">
17  </form>
```

```
10  <?php
11  |      echo $_POST["prenom"];
12  ?>
13
14  <form method="POST">
15  |      <input type="text" name="prenom">
16  |      <input type="submit">
17  </form>
```


Les tableaux superglobaux `$_GET` et `$_POST`

```
if($_GET) { // Si le formulaire a été soumis, j'affiche les données saisies  
    echo "Bienvenue {$_GET['prenom']} {$_GET['nom']} !";  
}
```

```
if($_POST) { // Si le formulaire a été soumis, j'affiche les données saisies  
    echo "Bienvenue {$_POST['prenom']} {$_POST['nom']} !";  
}
```



TP1 - Calculatrice

Les fonctions et la récupération de données

TP1 - Calculatrice - 5 opération mathématiques (1/3)

- Dans un nouveau fichier `calculator.php`, implémentez une **fonction** par opération mathématique (Cf. diapo 7) :
 - La **fonction** `add()` retourne l'addition de 2 **entiers** passés en **paramètre**
 - La **fonction** `subtract()` retourne la soustraction de 2 **entiers** passés en **paramètre**
 - La **fonction** `multiply()` retourne la multiplication de 2 **entiers** passés en **paramètre**
 - La **fonction** `divide()` retourne la division de 2 **entiers** passés en **paramètre**
 - La **fonction** `modulo()` retourne la modulo de 2 **entiers** passés en **paramètre**

TP1 - Calculatrice - La fonction `display()` (2/3)

- Ajoutez une **fonction** `display()` ne retourne pas de valeur mais se contente d'afficher une **chaîne de caractère** du type `"L'addition de <a> et donne <résultat>"`
- Cette **fonction** reçoit **2 entiers** en **paramètre** représentant les valeurs sur lesquelles réaliser l'opération mathématique
- Cette **fonction** reçoit un **3ème paramètre** représentant une **chaîne de caractères** correspondant à l'opération mathématique à réaliser (`"add"`, `"subtract"`, `"multiply"`, `"divide"` ou `"modulo"`)
- Proposer une 1ère version de la **fonction** `display()` en utilisant plusieurs blocs `if()` `elseif()`
- **Commentez** cette 1ère version de la fonction et proposez-en une 2nde en utilisant, cette fois, une **structure conditionnelle** de type `switch` afin faciliter l'algorithme

TP1 - Calculatrice - Les tests (3/3)

- Appelez 5 fois la fonction `display()` afin tester les 5 opérations mathématiques avec des valeurs **entières** différentes
- Voici une proposition d'affichage final attendu pour valider le TP :

L'addition de 2 et 3 donne **5**
La soustraction de 6 et 2 donne **4**
La multiplication de 4 et 3 donne **12**
La division de 4 et 2 donne **2**
Le modulo de 4 et 8 donne **4**

TP2 - Traiter les données d'un formulaire HTML

Les fonctions et la récupération de données

TP2 - Traiter les données d'un formulaire HTML (1/3)

- Dans un nouveau dossier, créez un fichier `form.php` contenant un **formulaire HTML** avec plusieurs **champs de saisie** :
 - Un champ de type `select` permettant de choisir le **genre** de l'utilisateur (`M.`, `Mme` ou `Autre`)
 - Un champ de type `text` permettant de saisir le **prénom** de l'utilisateur
 - Un champ de type `text` permettant de saisir le **nom** de l'utilisateur
 - Un champ de type `number` permettant de saisir l'**âge** de l'utilisateur
- Donnez une valeur à l'**attribut** `name` de chaque **champ** afin de pouvoir récupérer la donnée saisie simplement une fois le **formulaire** soumis
- La soumission du formulaire devra être gérée avec une **méthode** `GET` (Cf. [diapo 22](#))

TP2 - Traiter les données d'un formulaire HTML (2/3)

- S'il a été soumis par l'utilisateur, le formulaire ne doit plus apparaître à l'écran - Un message du type `"Bonjour M. Chris Chevalier - Vous avez 29 ans."` doit le remplacer
- Pour ce faire, assurez-vous que le formulaire a bien été soumis - Si c'est le cas, affichez les différentes données saisies à l'aide de la **variable superglobale** `$_GET` (Cf. [diapo 23](#))
- Si, au contraire, le **formulaire** n'a pas été soumis, affichez-le

TP2 - Traiter les données d'un formulaire HTML (3/3)

- Voici une proposition d'affichage final attendu pour valider le TP :

Genre

Prénom

Nom

Âge

Bonjour M. Chris Chevalier !
Vous avez 28 ans.



Bonus

Les fonctions et la récupération de données

★ Bonus

- 📅 Les règles de calcul des années bissextiles du calendrier grégorien indique qu'une année est bissextile tous les 4 ans donc si l'année est divisible par 4 (comme 1992 ou 1996, par exemple)
- ⚠ Mais ces règles ne suffisent pas ! En effet, lorsqu'il s'agit de la première année d'un nouveau siècle (1900, 2000 ou 2100, par exemple), cette année doit être divisible non plus par **4** mais par **400**
- 💡 Écrivez une **fonction PHP** `bissextiles()` qui retourne un **chaîne de caractère** affichant le nombre d'années bissextiles que vous avez vécu depuis votre année de naissance en indiquant lesquelles => `"Vous avez vécu 8 années bissextiles : 1992, 1996, 2000, 2004, 2008, 2012, 2016 et 2020"`

☆☆ Bonus II

- Modifiez la **fonction** `bissextiles()` afin qu'elle retourne une **chaîne de caractères** affichant toutes les années bissextiles entre l'année passée en **paramètre de l'URL** et l'année actuelle (`localhost:8888/index.php?annee=1992`)
- Si l'année passé en **paramètre de l'URL** est bissextile, affichez un message de type `"L'année 2000 est bissextiles !"`
- Voici une proposition d'affichage attendu : `"2000 est une années bissextile !
Tout comme 2004, 2008, 2012, 2016 et 2020"`

☆☆☆ Bonus III

- Rédigez un nouveau **script PHP** envoyant, via un formulaire HTML, une année sélectionnée dans une liste déroulante (`<select></select>`)
- La liste déroulante propose toutes les années de celle de votre naissance à l'actuelle
- Ce script affiche si l'année sélectionnée est bissextile ou non

2000 est une année bissextile !

2000 ▼

Envoyer

Bonus IV

- Entraînez vous seul, il existe de nombreux site avec des exercices pour progressez en **PHP**
- Si vous êtes plutôt joueur, regardez <https://www.codingame.com/>
- Si vous êtes plutôt progression pure et dure, lancez vous dans <https://www.codewars.com/>
- Dans les deux cas, parlez en à votre formateur pour plus de conseils

Bonus V

- Reprenez le **TP bonus** ★ présenté sur la **diapositive 59** du cours « **2 - Les bases du HTML** »
- Prévoyez une **page de contact** permettant aux visiteurs de votre site web d'envoyer un email aux administrateurs
- Valider la cohérence des informations saisies par l'utilisateur à l'aide de **fonctions** et de **conditions** PHP :
 - L'email saisi respecte bien le format type d'un email ?
 - Les nom et prénom ne sont pas vides ou ne font pas plus de 80 caractères ?
 - Le message n'est pas vide et ne fait pas moins de 10 caractères et pas plus 500 ?
 - etc.

Des questions ?