Chapitre 3
Utiliser les fonctions PHP

1. Préambule

L'objectif de ce chapitre est de présenter les fonctions les plus utiles dans le cadre du développement d'un site web.

PHP propose de nombreuses fonctions ; la description de chaque fonction est accessible en ligne sur le site www.php.net.

Version 8

Depuis la **version 8**, il est possible de passer des paramètres à une fonction en utilisant le nom du paramètre au lieu de sa position. Cette fonctionnalité est présentée dans le chapitre Écrire des fonctions et des classes PHP, mais elle peut être utilisée pour les fonctions natives du langage PHP, et donc pour les fonctions présentées dans ce chapitre. Par contre, dans ce chapitre, les noms réels des paramètres des fonctions ne sont pas présentés (ils sont traduits) ; pour les connaître, consultez la documentation en ligne des fonctions.

Depuis la **version 8.1**, passer la valeur `NULL` à un paramètre qui n'est pas explicitement optionnel est déprécié et génère donc une alerte de niveau `E_DEPRECATED`.

Exemple

```
<?php
$x = null;
$n = strlen($x);
?>
```

Résultat

```
Deprecated: strlen(): Passing null to parameter #1 ($string) of type string
is deprecated in /app/scripts/index.php on line 3
```

2. Manipuler les constantes, les variables et les types de données

2.1 Constantes

PHP propose un certain nombre de fonctions utiles sur les constantes :

Nom	Rôle
`defined`	Indique si une constante est définie ou non.
`constant`	Retourne la valeur d'une constante.

defined

La fonction `defined` permet de savoir si une constante est définie ou non.

Syntaxe

```
booléen defined(chaîne nom)
```

`nom` Nom de la constante.

La fonction `defined` retourne `TRUE` si la constante est définie et `FALSE` dans le cas contraire.

Exemple

```
<?php
// Tester si la constante CONSTANTE est définie.
$ok = defined('CONSTANTE');
if ($ok) {
  echo 'CONSTANTE est définie.<br />';
} else {
  echo 'CONSTANTE n\'est pas définie.<br />';
};
// Définir la constante CONSTANTE
define('CONSTANTE','valeur de CONSTANTE');
// Tester si la constante CONSTANTE est définie.
$ok = defined('CONSTANTE');
if ($ok) {
  echo 'CONSTANTE est définie.<br />';
} else {
  echo 'CONSTANTE n\'est pas définie.<br />';
};
?>
```

Résultat

```
CONSTANTE n'est pas définie.
CONSTANTE est définie.
```

constant

La fonction `constant` retourne la valeur d'une constante dont le nom est passé en paramètre.

Syntaxe

```
mixte constant(chaîne nom)
```

Avec :

`nom` Nom de la constante.

Cette fonction est pratique pour récupérer la valeur d'une constante dont le nom n'est pas connu a priori.

Exemple

```
<?php
// définir le nom de la constante dans une variable
$nomConstante = 'AUTRE CONSTANTE';
// définir la valeur de la constante
define($nomConstante,'valeur de AUTRE CONSTANTE');
// afficher la valeur de la constante
echo $nomConstante,' = ',constant($nomConstante);
?>
```

Résultat

```
AUTRE CONSTANTE = valeur de AUTRE CONSTANTE
```

D'autres fonctions permettent de connaître le type d'une constante (cf. section Manipuler les constantes, les variables et les types de données - Types de données).

2.2 Variables

PHP propose un certain nombre de fonctions utiles sur les variables :

Nom	Rôle
`empty`	Indique si une variable est vide ou non.
`isset`	Indique si une ou plusieurs variables sont définies ou non.
`unset`	Supprime une ou plusieurs variables.
`var_dump`	Affiche des informations sur une ou plusieurs variables (type et valeur).

empty

La fonction `empty` permet de tester si une variable est vide ou non.

Syntaxe

```
booléen empty(mixte variable)
```

`variable` Variable à tester.

`empty` retourne `TRUE` si la variable est vide et `FALSE` dans le cas contraire.

Une variable est considérée comme vide si elle n'a pas été affectée ou si elle contient une chaîne vide (""), une chaîne égale à 0 ("0"), un 0, `NULL`, `FALSE` ou un tableau vide.

La fonction `empty` peut aussi être utilisée pour tester si une expression est vide ou non.

Exemple

```
<?php
// Test d'une variable non initialisée.
$est_vide = empty($variable);
echo '$variable non initialisé<br />';
if ($est_vide) {
  echo '=> $variable est vide.<br />';
} else {
  echo '=> $variable n\'est pas vide.<br />';
}
// Test d'une variable contenant une chaîne vide.
$variable = '';
$est_vide = empty($variable);
echo '$variable = \'\'<br />';
if ($est_vide) {
  echo '=> $variable est vide.<br />';
} else {
  echo '=> $variable n\'est pas vide.<br />';
}
// Test d'une variable contenant une chaîne égale à 0.
$variable = '0';
$est_vide = empty($variable);
echo '$variable = \'',$variable,'\'<br />';
if ($est_vide) {
  echo '=> $variable est vide.<br />';
} else {
  echo '=> $variable n\'est pas vide.<br />';
}
// Test d'une variable contenant 0.
$variable = 0;
$est_vide = empty($variable);
echo '$variable = ',$variable,'<br />';
```

```
if ($est_vide) {
  echo '=> $variable est vide.<br />';
} else {
  echo '=> $variable n\'est pas vide.<br />';
}
// Test d'une variable contenant une chaîne non vide.
$variable = 'x';
$est_vide = empty($variable);
echo '$variable = \'',$variable,'\'<br />';
if ($est_vide) {
  echo '=> $variable est vide.<br />';
} else {
  echo '=> $variable n\'est pas vide.<br />';
}
?>
```

Résultat

```
$variable non initialisé
=> $variable est vide.
$variable = ''
=> $variable est vide.
$variable = '0'
=> $variable est vide.
$variable = 0
=> $variable est vide.
$variable = 'x'
=> $variable n'est pas vide.
```

isset

La fonction `isset` permet de tester si une ou plusieurs variables sont définies ou non.

Syntaxe

```
booléen isset(mixte variable[, ...])
```

`variable` Variable à tester (éventuellement plusieurs, séparées par une virgule).

`isset` retourne `TRUE` si la variable est définie et `FALSE` dans le cas contraire.

Si plusieurs paramètres sont fournis, la fonction retourne `TRUE` uniquement si toutes les variables sont définies.

Une variable est considérée comme non définie si elle n'a pas été affectée ou si elle contient `NULL`. À la différence de la fonction `empty`, une variable qui contient une chaîne vide (""), une chaîne égale à 0 ("0"), un 0, un `FALSE` ou un tableau vide, n'est pas considérée comme non définie.

Exemple

```
<?php
// Test d'une variable non initialisée.
$est_définie = isset($variable);
echo '$variable non initialisé<br />';
if ($est_définie) {
  echo '=> $variable est définie.<br />';
} else {
  echo '=> $variable n\'est pas définie.<br />';
}
// Test d'une variable contenant une chaîne vide.
$variable = '';
$est_définie = isset($variable);
echo '$variable = \'\'<br />';
if ($est_définie) {
  echo '=> $variable est définie.<br />';
} else {
  echo '=> $variable n\'est pas définie.<br />';
}
// Test d'une variable contenant une chaîne égale à 0.
$variable = '0';
$est_définie = isset($variable);
echo '$variable = \'',$variable,'\'<br />';
if ($est_définie) {
  echo '=> $variable est définie.<br />';
} else {
  echo '=> $variable n\'est pas définie.<br />';
}
// Test d'une variable contenant 0.
$variable = 0;
$est_définie = isset($variable);
echo '$variable = ',$variable,'<br />';
if ($est_définie) {
  echo '=> $variable est définie.<br />';
} else {
  echo '=> $variable n\'est pas définie.<br />';
}
// Test d'une variable contenant une chaîne non vide.
$variable = 'x';
$est_définie = isset($variable);
echo '$variable = \'',$variable,'\'<br />';
if ($est_définie) {
  echo '=> $variable est définie.<br />';
} else {
  echo '=> $variable n\'est pas définie.<br />';
}
?>
```

Résultat

```
$variable non initialisé
=> $variable n'est pas définie.
$variable = ''
=> $variable est définie.
$variable = '0'
=> $variable est définie.
$variable = 0
=> $variable est définie.
$variable = 'x'
=> $variable est définie.
```

unset

La fonction `unset` permet de supprimer une ou plusieurs variables.

Syntaxe

```
unset(mixte variable[, ...])
```

`variable` Variable à supprimer (éventuellement plusieurs, séparées par une virgule).

Après suppression, la variable se trouve dans le même état que si elle n'avait jamais été affectée. L'utilisation de la fonction `isset` sur une variable supprimée retourne `FALSE` notamment.

Exemple

```
<?php
// Définir une variable.
$variable = 1;
// Afficher la variable et tester si elle est définie.
$est_définie = isset($variable);
echo '$variable = ',$variable,'<br />';
if ($est_définie) {
  echo '=> $variable est définie.<br />';
} else {
  echo '=> $variable n\'est pas définie.<br />';
}
// Supprimer la variable.
unset($variable);
// Afficher la variable et tester si elle est définie.
$est_définie = isset($variable);
echo '$variable = ',$variable??'','<br />';
if ($est_définie) {
  echo '=> $variable est définie.<br />';
} else {
  echo '=> $variable n\'est pas définie.<br />';
}
?>
```

Résultat

```
$variable = 1
=> $variable est définie.
$variable =
=> $variable n'est pas définie.
```

Remarque

Affecter un 0 ou une chaîne vide à une variable ne la supprime pas.

var_dump

La fonction `var_dump` affiche des informations sur une ou plusieurs variables (type et contenu).

Syntaxe

```
var_dump(mixte variable[, ...])
```

`variable` Variable à afficher (éventuellement plusieurs, séparées par une virgule).

La fonction `var_dump` est surtout intéressante lors des phases de mise au point.

Exemple

```
<?php
// afficher les informations sur une variable non initialisée
$variable = NULL;
var_dump($variable);
// initialiser la variable avec un nombre entier
$variable = 10;
// afficher les informations sur la variable
echo '<br />';
var_dump($variable);
// modifier la valeur (et le type) de la variable
$variable = 3.14; // nombre décimal
// afficher les informations sur la variable
echo '<br />';
var_dump($variable);
// modifier la valeur (et le type) de la variable
$variable = 'abc'; // chaîne de caractères
// afficher les informations sur la variable
echo '<br />';
var_dump($variable);
?>
```

Chapitre 2
Gestion des utilisateurs

Durée : 3 heures 35

Mots-clés

Code PHP, formulaire, requêtes HTTP, service web, base de données, MySQL, phpMyAdmin.

Objectifs

C'est vers la création des premières pages PHP avec accès à la base de données que sont tournés les prochains TP. Ce chapitre n'introduit pas de framework PHP, mais inscrit néanmoins le développement dans un écosystème auquel participent d'autres technologies telles que Node.js ou HTML 5.

Prérequis

Pour valider les prérequis nécessaires, avant d'aborder le TP, répondez aux questions ci-après :

1. Quel est l'attribut de la balise `<form>` qui détermine l'URL de traitement des données d'un formulaire ?
   - **a.** `process`
   - **b.** `data-process`
   - **c.** `action`
   - **d.** `react`
2. Que signifie la valeur `POST` de l'attribut `method` ?
   - **a.** Les données sont envoyées au serveur par mail.
   - **b.** Les données sont envoyées sous forme d'une liste clé-valeur dans la query string de l'URL.
   - **c.** Les données sont signées.
   - **d.** Les données sont envoyées sous forme d'une liste clé-valeur dans le corps de la requête HTTP.

3. Quelle est la variable PHP qui reprend les données postées d'un formulaire ?

- **a.** `$_SERVER`
- **b.** `$_POST`
- **c.** `$_FORM`
- **d.** `$_QUERY`
- **e.** `$_FORM`

4. Quelles sont les principales fonctions de PHP pour exécuter une requête MySQL ?

- **a.** `db_mysql_connect`, `db_mysql_query`, `db_mysql_close`
- **b.** `mysql_connect`, `mysql_query`, `mysql_fetch`, `mysql_close`
- **c.** `php_mysql_query`, `php_mysql_fetch_array`

5. Selon le standard REST, quel est le verbe HTTP pour un appel de service web avec paramètres retournant des données ?

- **a.** `GET`
- **b.** `POST`
- **c.** `GETLIST`
- **d.** `HEAD`

6. Quels sont les formats en sortie généralement produits par un service web REST ?

- **a.** XML
- **b.** JSON
- **c.** PHP
- **d.** HTML

7. Comment s'appelle l'utilitaire de base de données pour MySQL écrit en PHP ?

- **a.** MySQLMyPHP
- **b.** Maria Admin
- **c.** phpMyAdmin
- **d.** MyPHPAdmin

Corrigé p. 115

Énoncé 2.1 Création de la base de données MySQL

Durée estimative : 30 minutes
La première étape dans la réalisation de notre site dynamique consiste à créer la base de données et sa première table `utilisateur'.

- Lancez phpMyAdmin et connectez-vous au serveur local.
- Créez un compte d'utilisateur app_teamup.
- Créez la base de données `teamup` avec un classement de caractère insensible à la casse.
- Créez la table `utilisateur` formée des colonnes suivantes :

Colonne	Attributs de la colonne
`id_utilisateur`	Int, auto_increment, clé primaire
`utilisateur_nom`	Varchar(100)
`utilisateur_login`	Varchar(100)
`utilisateur_pwd`	Varchar(100)
`utilisateur_email`	Varchar(100)
`utilisateur_creation`	Datetime, peut être NULL

- Insérez dans la table deux enregistrements.
- Sélectionnez le contenu de la table

Corrigé p. 116

Énoncé 2.2 Saisie d'un utilisateur avec un template HTML simple

Durée estimative : 45 minutes
Les choses sérieuses commencent ! Ce TP met en place le code PHP pour ajouter des utilisateurs dans la base de données. Même s'il ne repose pas sur la logique MVC, la structure du code suit pourtant un découpage entre différentes préoccupations : présentation, traitement, service et accès aux données.

- Ajoutez dans le répertoire **application** un dossier **dal** (pour Data Access Layer) et dans ce dernier un fichier **userdao.php** contenant une classe vide `UserDAO`.
- Ajoutez dans le répertoire **application** un dossier **service** ainsi qu'un fichier **userservice.php** contenant une classe vide `UserService`.

- Ajoutez dans le répertoire **application/models** un fichier **userentity.php** contenant une classe `UserEntity`. Définissez dans cette classe un champ pour chaque colonne de la table `utilisateur`.
- Ajoutez dans le répertoire **application** un fichier **adduser.php**. Copiez-collez le contenu du fichier **views/_layout.php**, retirez le contenu de la section `<body>` tout en conservant la barre de navigation. Vous devez inclure le fichier de configuration **config.php**.
- Installez un formulaire « auto-postant » avec des champs pour saisir chaque colonne de la table `utilisateur` (sauf la date de création). La clé `id_utilisateur` est représentée par un champ caché. Prévoyez un bouton de soumission du formulaire.
- Ajoutez dans le fichier de configuration des propriétés globales pour se connecter à la base de données MySQL : `$cx_server`, `$cx_login`, `$cx_pwd`, `$cx_dbname`. Définissez dans ce même fichier config.php une fonction globale `get_default_connection()` qui retourne l'ensemble des propriétés de connexion à la base de données.
- Ajoutez dans la classe `UserDAO` un constructeur. Le constructeur doit récupérer les paramètres de connexion depuis la classe `get_default_connection()` et la stocker dans une propriété privée `db_connection`.
- Ajoutez dans la classe `UserDAO` une méthode `adduser($userentity)`. Cette méthode insère dans la table `utilisateur` un enregistrement dont les valeurs de champs sont portées par le paramètre `$userentity`.
- Ajoutez dans la classe `UserService` une méthode `adduser($userentity)`. Cette méthode instancie la classe `UserDAO` et invoque la méthode `adduser`.
- Dans le fichier **utilisateur.php**, aménagez un script PHP et testez si la requête HTTP est de type `POST`. Vous devez extraire les données du formulaire postées et initialiser une instance de `UserEntity`. Instanciez la classe `UserService` et appelez la méthode `adduser`.
- Testez le formulaire, saisissez des valeurs et cliquez sur le bouton de soumission des données. Vérifiez dans la base de données la présence du nouvel enregistrement.

Corrigé p. 120

Énoncé 2.3 Affichage d'un utilisateur avec template réactif (responsive)

Durée estimative : 40 minutes
Cet exercice associe PHP et Bootstrap pour présenter la liste des utilisateurs dans un template réactif. Le code PHP est employé pour effectuer une requête de sélection auprès de la base de données et le framework Bootstrap apporte ses composants d'affichage optimisé pour différents équipements (Web, mobile…).

- Ajoutez dans la classe `UserDAO` la méthode `getuserlist($filtrenom=null)`. Comme dans le TP précédent, récupérez la connexion depuis la propriété `db_connection`. Formez deux requêtes SQL pour sélectionner les enregistrements de la table utilisateur, selon que le paramètre de la méthode `$filtrenom` est null ou non. S'il est null, la requête sélectionne la totalité des enregistrements de la table `utilisateur`. Si le paramètre `$filtrenom` est non null, ajoutez une condition sur la colonne `utilisateur_nom` dans la requête.

Avant d'utiliser l'opérateur SQL `like`, vérifiez que le paramètre `$filtrenom` ne comporte pas de signes pouvant dénaturer la requête SQL et ainsi exposer ce code à des attaques par injection de script SQL. Les signes interdits sont notamment le point-virgule et les apostrophes.

  Instanciez pour chaque enregistrement un objet `UserEntity` et retournez un tableau indexé contenant la collection d'objets obtenue à partir de la lecture de la table `utilisateur`.

- Ajoutez dans `UserService` une méthode `getuserlist($filtrenom)` chargée d'appeler `UserDAO::getuserlist($filtrenom)`.
- Créez une page application/utilisateurs.php et changez dans le fichier **menu.json** la propriété `route='#'` en `route='utilisateurs.php'` pour l'entrée située à l'index 1.

  Reprenez dans utilisateurs.php la mise en page générale depuis _layout.php et testez la page.
- Aménagez un script PHP, instanciez `UserService` et appelez la méthode `getuserlist` sans paramètre.
- Itérez à l'aide d'une boucle `foreach` dans la collection d'objets `UserEntity` et affichez les propriétés `utilisateur_id`, `utilisateur_nom`, `utilisateur_email` dans un tableau.

  Ajoutez les attributs Bootstrap pour rendre le tableau réactif. Testez le tableau.
- Dans un formulaire, ajoutez un champ texte ayant pour identifiant `filtrenom` afin de filtrer la liste à partir du nom. Le formulaire doit aussi comporter un bouton pour soumettre le formulaire.

- Définissez les propriétés du formulaire sur `POST` et utilisateurs.php. Modifiez le script PHP pour appeler la méthode `getuserlist($filtrenom)` si le verbe est un `POST`.

  Testez le filtrage de la liste avec des préfixes de nom.

- Ajoutez un lien vers la page adduser.php.

Corrigé p. 125

Énoncé 2.4 Gestion des équipes

Durée estimative : 60 minutes

Ce TP est en quelque sorte la combinaison des deux précédents. Il s'agit de réaliser l'interface de gestion des équipes d'utilisateurs en reprenant la même logique de séparation du code PHP en plusieurs couches. Patience, ce travail n'est pas superflu et trouvera très vite son utilité quand le site sera plus étoffé.

- Ajoutez dans la base de données les tables `equipe(id_equipe,equipe_nom)` et `utilisateur_equipe(id_equipe,id_utilisateur)`.
- Créez dans Eclipse la classe `TeamEntity` reprenant chaque colonne de la table `equipe` sous la forme d'un champ.
- Ajoutez au projet deux classes `TeamService` et `TeamDAO` comportant les méthodes `getteamlist()`, `addteam($teamentity)`, `editteam($teamentity)`. Employez des requêtes `SELECT`, `INSERT` et `UPDATE` pour implémenter ces trois méthodes.
- Ajoutez les méthodes `getuserteam($id_equipe)`, `adduserteam($id_utilisateur, $id_equipe)`, `removeuserteam($id_utilisateur, $id_equipe)`. Utilisez des requêtes `SELECT`, `INSERT` et `DELETE` pour implémenter ces trois méthodes.

  Définissez également la méthode `getusernotinteam($id_equipe)` qui sélectionne les utilisateurs n'appartenant pas à une équipe.

- Dans le répertoire **application**, ajoutez la page **equipes.php** en reprenant le contenu du fichier **_layout.php**. Installez un lien dans la page **utilisateurs.php** pour naviguer vers **equipes.php**.
- Divisez l'écran **equipes.php** en colonnes : d'abord une zone de texte et une liste pour ajouter et afficher les équipes, puis encore deux listes pour afficher la composition d'une équipe et les utilisateurs n'appartenant pas à cette équipe.

  Chargez la liste des équipes avec la classe `TeamService`.

  Ajoutez un bouton de soumission `cmd_addteam` pour créer une nouvelle équipe.

  Dans le code `POST`, appelez la méthode `addteam()` et rafraîchissez le contenu de la liste des équipes.