Editions ENI

PHP 7

Développez un site web dynamique et interactif

(2e édition)

Collection
Ressources Informatiques

Extrait

Chapitre 6
Gérer les formulaires et les liens

1. Vue d'ensemble

1.1 Introduction

Dans les sites web dynamiques, il est très souvent nécessaire d'interagir avec l'utilisateur.

En HTML, il existe principalement deux méthodes pour interagir avec un utilisateur :

– les liens (balise `<a>`) ;

– les formulaires (balise `<form>`).

Des scripts PHP peuvent être utilisés pour traiter le clic de l'utilisateur sur un lien ou la saisie de l'utilisateur dans un formulaire.

1.2 Les liens

Le lien est la technique de base qui permet à un utilisateur de naviguer entre les différentes pages d'un site.

Un lien HTML est défini entre les balises `<a>` et `</a>`.

Syntaxe simplifiée

```
<a
  [ href="url" ]
  [ id="identifiant_lien" ]
  [ target="cible" ]
```

```
>
...
</a>
```

Les attributs de la balise <a> sont les suivantes :

`href` URL (*Uniform Resource Locator*) relative ou absolue qui est appelée par le lien.

`id` Identifiant du lien. Si la page HTML contient plusieurs liens, l'identifiant permet de les différencier. En ce qui nous concerne, cet identifiant ne présente pas d'intérêt car il n'est pas récupéré dans le script de traitement du lien. Par contre, il peut être utilisé côté client, en JavaScript par exemple.

`target` Cible (par exemple une autre fenêtre) dans laquelle ouvrir l'URL cible. Par défaut, l'URL cible s'affiche dans la même fenêtre.

L'URL peut contenir des paramètres qui permettent de passer des informations d'une page à une autre.

Syntaxe

```
url_classique?nom=valeur[&...]
```

Le point d'interrogation (?) introduit la liste des paramètres de l'URL séparés par le caractère esperluette (&) ; chaque paramètre est constitué par un couple nom/valeur sous la forme `nom=valeur` :

```
www.monsite.com/info/accueil.php?prenom=Olivier
chercher.php?prenom=Olivier&nom=HEURTEL
```

Exemple

– Script `page1.php`

```
<?php
// Initialisation d'une variable.
$nom='Olivier';
?>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="fr">
  <head><meta charset="utf-8" /><title>Page 1</title></head>
  <body>
    <div>
    <!-- lien vers la page 2 en passant la valeur de $nom
         dans l'URL -->
    <a href="page2.php?nom=<?= $nom ?>">Page 2</a>
    </div>
  </body>
</html>
```

– Source de la page dans le navigateur

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="fr">
  <head><meta charset="utf-8" /><title>Page 1</title></head>
  <body>
    <div>
    <!-- lien vers la page 2 en passant la valeur de $nom
         dans l'URL -->
    <a href="page2.php?nom=Olivier">Page 2</a>
    </div>
  </body>
</html>
```

Résultat

– Affichage de la page 1 :

– Résultat du clic sur le lien :

Pour l'instant, aucun nom n'est affiché dans la deuxième page. La variable `$nom` définie dans le script `page1.php` n'est pas disponible dans le script `page2.php` (voir le chapitre Introduction à PHP, section Les bases du langage PHP - Variables - Portée et durée de vie). De plus, notre script ne contient aucune instruction permettant de récupérer les données passées dans l'URL ; nous verrons comment procéder à la section Récupérer les données d'une URL ou d'un formulaire.

1.3 Les formulaires

1.3.1 Petit rappel sur les formulaires

Le formulaire est un outil de base indispensable pour les sites web dynamiques puisqu'il permet à l'utilisateur de saisir des informations et donc d'interagir avec le site.

Un formulaire HTML est défini entre les balises `<form>` et `</form>`.

Syntaxe simplifiée

```
<form
  [ action="url_de_traitement" ]
  [ method="GET"|"POST" ]
  [ id="identifiant_formulaire" ]
  [ target="cible" ]>

...
</form>
```

Les attributs de la balise `<form>` sont les suivants :

`action` URL (*Uniform Resource Locator*) relative ou absolue qui va traiter le formulaire (en ce qui nous concerne, un script PHP). Cet attribut est obligatoire pour se conformer à la recommandation XHTML stricte.

`method` Mode de transmission vers le serveur des informations saisies dans le formulaire.

`GET` (valeur par défaut) : les données du formulaire sont transmises dans l'URL.

`POST` : les données du formulaire sont transmises dans le corps de la requête.

`id` Identifiant du formulaire. Si la page HTML contient plusieurs formulaires, l'identifiant permet de les différencier. En ce qui nous concerne, cet identifiant ne présente pas d'intérêt car il n'est pas récupéré dans le script de traitement du formulaire. Par contre, il peut être utilisé côté client, en JavaScript par exemple.

`target` Cible (par exemple une autre fenêtre) dans laquelle ouvrir l'URL cible. Par défaut, l'URL cible s'affiche dans la même fenêtre.

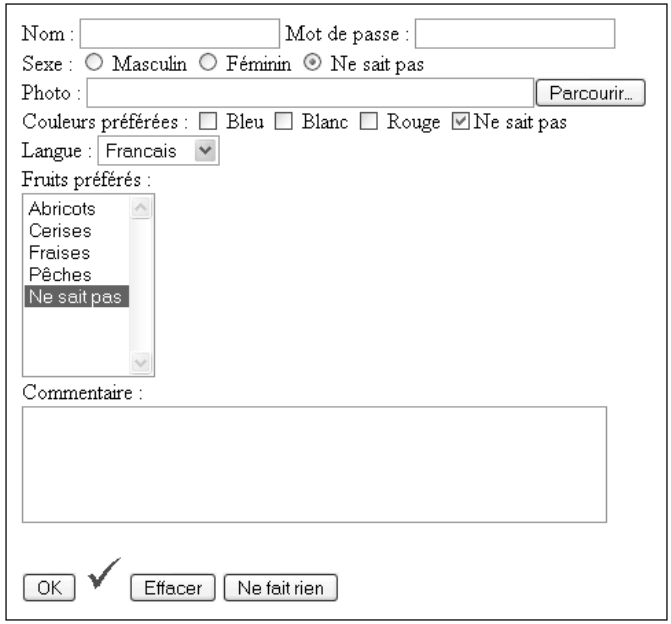
Entre les balises `<form>` et `</form>`, il est possible de placer des balises `<input>`, `<select>` ou `<textarea>` pour définir des zones de saisie.

Exemple (formulaire HTML complet)

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="fr">
  <head>
    <meta charset="utf-8" />
    <title>Saisie</title>
  </head>
  <body>
    <form action="saisie.php" method="post">
    <div>
      Nom :
      <input type="text" name="nom" value=""
           size="20" maxlength="20" />
      Mot de passe :
      <input type="password" name="mot_de_passe" value=""
           size="20" maxlength="20" />
      <br />Sexe :
      <input type="radio" name="sexe" value="M" />Masculin
      <input type="radio" name="sexe" value="F" />Féminin
      <input type="radio" name="sexe" value="?"
                          checked="checked" />Ne sait pas
      <br />Photo :
      <input type="file" name="photo" size="50" />
      <br />Couleurs préférées :
      <input type="checkbox" name="couleurs[bleu]" />Bleu
      <input type="checkbox" name="couleurs[blanc]" />Blanc
      <input type="checkbox" name="couleurs[rouge]" />Rouge
      <input type="checkbox" name="couleurs[pas]"
                               checked="checked" />Ne sait pas
      <br />Langue :
      <select name="langue">
        <option value="E">Espagnol</option>
        <option value="F" selected="selected" >Francais</option>
        <option value="I">Italien</option>
      </select>
      <br />Fruits préférés :<br />
      <select name="fruits[]" multiple="multiple" size="8">
        <option value="A">Abricots</option>
        <option value="C">Cerises</option>
        <option value="F">Fraises</option>
        <option value="P">Pêches</option>
        <option value="?" selected="selected">
                         Ne sait pas</option>
      </select>
      <br />Commentaire :<br />
      <textarea name="commentaire" rows="4" cols="50"></textarea>
      <br />
      <input type="hidden" name="invisible" value="123" /><br />
```

```
      <input type="submit" name="soumettre" value="OK" />
      <input type="image" name="valider" alt="valider"
src="valider.gif" />
      <input type="reset" name="effacer" value="Effacer" />
      <input type="button" name="action" value="Ne fait rien" />
    </div>
    </form>
  </body>
</html>
```

Résultat



PHP peut intervenir à deux endroits par rapport au formulaire :

– Pour la construction du formulaire, si ce dernier doit contenir des informations dynamiques.
– Pour le traitement du formulaire (c'est-à-dire des données saisies par l'utilisateur dans le formulaire).

Editions ENI

MySQL 8
Administration et optimisation

Collection
Ressources Informatiques

Extrait

Chapitre 4
Sécurité et gestion des utilisateurs

1. Introduction

Savez-vous qu'une bonne partie des attaques informatiques les plus médiatisées concernent les bases de données ? En y réfléchissant, c'est parfaitement logique : les parties les plus sensibles d'un système informatique sont les données personnelles (identifiants, mots de passe, nationalité, salaire, historique de santé…) qui sont stockées dans une base de données. Il est donc impératif de veiller à la sécurisation de vos serveurs MySQL si vous voulez éviter la mauvaise publicité liée à une fuite de données.

Ce chapitre a pour but de vous donner les clés pour assurer la sécurité de votre système MySQL. Notez que nous nous limitons aux éléments spécifiques à MySQL et que d'autres éléments tout aussi importants tels que la sécurisation du serveur hôte ou du réseau ne sont pas abordés.

2. Sécurisation du serveur

Les questions liées à la sécurité doivent être posées dès l'installation du serveur de base de données. Dans la philosophie de MySQL l'utilisateur doit pouvoir télécharger, installer et utiliser le SGBDR (système de gestion de bases de données relationnelles) sans aucune difficulté. Cette simplicité est l'un des points forts de MySQL, car d'une part cela a permis de démocratiser l'utilisation d'une base de données en rendant accessible cet univers (beaucoup de développeurs ont connu les bases de données grâce à MySQL) et d'autre part, cela donne la possibilité de tester très simplement son application. Mais historiquement, l'installation par défaut a longtemps été beaucoup trop permissive, ce qui explique pourquoi on trouve encore nombre de serveurs mal sécurisés. De gros progrès ont été réalisés à partir notamment de MySQL 5.7.

2.1 Sécurisation de l'installation

Le but de cette section n'est pas de revenir sur l'installation du serveur qui est détaillée au chapitre Installation du serveur, mais simplement de rappeler quelques points cruciaux concernant la sécurité. Vous devrez certainement les adapter en fonction de votre type d'installation et de votre système d'exploitation.

2.1.1 Contrôler les droits

Vous devez créer un groupe et un utilisateur dédié pour lancer l'instance `mysqld` (cette étape est effectuée automatiquement si vous utilisez un installeur ou votre gestionnaire de paquets) :

```
$ groupadd mysql
$ useradd -g mysql mysql
```

Le répertoire de données contient les informations sensibles, il doit donc être préservé d'actes de malveillance ou de la visualisation par des personnes non autorisées :

```
$ cd /usr/local/mysql/
$ chown -R root .
$ chown -R mysql data
$ chgrp -R mysql .
```

`mysqld` ne doit en aucun cas être exécuté avec l'administrateur système `root` sous UNIX (ou `administrateur` sous MS Windows). Un utilisateur avec par exemple le droit `FILE` pourrait créer des fichiers en tant que `root`.

2.1.2 Ajouter un mot de passe au compte utilisateur root

C'est le superutilisateur de MySQL (à ne pas confondre avec l'administrateur système `root` sous UNIX) ; il a donc tous les droits sur le serveur ; il doit être protégé par un mot de passe. Ceci est valable quel que soit le système d'exploitation.

```
$ mysql –u root # connexion au serveur avec l'utilisateur root sans
mot de passe
mysql> ALTER USER root@localhost IDENTIFIED BY 'm0T2p4ss3';
```

Comme vous allez le voir un peu plus loin, un utilisateur MySQL est composé d'un nom « user » et du nom de la machine à partir de laquelle l'utilisateur se connecte « host ». Vous pouvez donc avoir un compte `'root'@'localhost'` qui permet de se connecter au serveur avec l'utilisateur `root` à condition que le client soit sur la même machine que le serveur MySQL (en local) et par exemple un compte `'root'@'123.45.67.89'` qui, lui ne permet de se connecter en `root` qu'à partir de la machine qui a comme adresse IP 123.45.67.89. Ce sont bien deux comptes différents, qui peuvent avoir des droits et des mots de passe différents. Cette possibilité offerte par le serveur peut permettre une gestion des droits très fine. Cependant, par expérience, nous vous conseillons de n'avoir qu'une seule occurrence par utilisateur, par souci de simplification de la gestion des droits et donc pour diminuer les risques d'erreurs. En d'autres termes, ne gardez que l'utilisateur `'root'@'localhost'`. Si vous avez besoin d'administrer votre serveur à distance, au lieu d'avoir `'root'@'%'` (qui permet de se connecter avec l'utilisateur `root` depuis n'importe quelle machine), utilisez le protocole de communication sécurisée SSH ou un équivalent.

Exemple : Si vous trouvez plusieurs comptes `root` sur votre système, alors que seul le compte `root@localhost` est utile

```
mysql> SELECT user, host FROM mysql.user WHERE user = 'root' \G
*************************** 1. row ***************************
user: root
host: localhost
```

```
*************************** 2. row ***************************
user: root
host: 123.456.78.9

Vous pouvez supprimer le compte inutile :
mysql> DROP USER 'root'@'123.456.78.9' ;

mysql> SELECT user, host FROM mysql.user WHERE user = 'root' \G
*************************** 1. row ***************************
user: root
host: localhost
```

Remarque

Renommer le compte administrateur : vous pouvez renommer votre compte administrateur pour qu'il soit plus difficile à trouver par une personne malveillante : `RENAME USER 'root'@'localhost' TO 'leader'@'localhost';`

2.1.3 Supprimer les comptes anonymes

Des comptes anonymes, c'est-à-dire des comptes ayant le champ `user` à vide, peuvent être présents sur votre système. Certains installeurs créaient automatiquement un tel compte dans le passé avec certaines versions de MySQL. Un compte anonyme permet de se connecter au serveur avec n'importe quel utilisateur, en local et sans mot de passe : pratique, mais franchement problématique au niveau sécurité. Mieux vaut supprimer au plus vite ce genre de comptes, qui peuvent être trouvés avec la requête suivante :

```
mysql> SELECT user, host FROM mysql.user WHERE user = '' \G
*************************** 1. row ***************************
user:
host: localhost
password:
```

Remarque

Depuis MySQL 5.7, plus aucun compte anonyme n'est créé lors de l'installation du serveur.

2.1.4 Supprimer le schéma test

Le schéma `test` est créé par MySQL lors de l'installation pour toutes les versions avant 5.7 et tous les utilisateurs ont le droit d'y accéder en lecture comme en écriture sans qu'il soit nécessaire de spécifier explicitement les droits adéquats. Un utilisateur mal intentionné pourrait alors remplir de données votre disque dur et ainsi empêcher le bon fonctionnement de votre application. Si ce schéma n'a aucune utilité pour votre application, mieux vaut le supprimer :

```
mysql> SHOW SCHEMAS;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| test               |
+--------------------+

mysql> DROP SCHEMA test;
```

À noter que cette recommandation s'applique également à tout schéma `test` qui serait créé par la suite, mais également à tout schéma qui commencerait par le mot `test_`.

2.1.5 Sécuriser votre installation avec l'outil mysql_secure_installation

La mise en œuvre de ces différentes recommandations peut être effectuée avec l'outil `mysql_secure_installation`. Cet outil n'est pas disponible sous MS Windows, cependant l'installeur graphique propose également de sécuriser l'installation.

Il suffit de lancer le script en tant qu'utilisateur Linux `root` et de suivre les indications à l'écran :

```
# mysql_secure_installation
Securing the MySQL server deployment.

Connecting to MySQL using a blank password.

VALIDATE PASSWORD PLUGIN can be used to test passwords
and improve security. It checks the strength of password
```

```
and allows the users to set only those passwords which are
secure enough. Would you like to setup VALIDATE PASSWORD plugin?
Press y|Y for Yes, any other key for No:
```

2.2 Chiffrement des données

Pour faire face aux besoins de chiffrement d'informations sensibles (financières, médicales...) ou pour tenter de se prémunir contre le vol de données, contre des administrateurs peu scrupuleux, il peut être nécessaire de crypter ses données. Avant la version 5.7, MySQL ne possédait pas de modules de chiffrement capables par exemple de crypter le contenu d'une colonne ou d'une table. Une possibilité était cependant de crypter/décrypter des données avec les fonctions MySQL telles que `AES_ENCRYPT()` / `AES_DECRYPT()`, `DES_ENCRYPT()` / `DES_DECRYPT()` et `ENCODE()` / `DECODE()`.

Remarque

Pour plus de détails, consultez la page suivante :
https://dev.mysql.com/doc/refman/8.0/en/encryption-functions.html

Les données sont alors cryptées sur le disque. Par contre, elles sont en clair, y compris la clé de cryptage dans le code SQL ainsi que sur le réseau. Par exemple :

```
mysql> INSERT INTO t_crypt VALUES (1, AES_ENCRYPT('Phrase cryptée',
'crypt_key'));

mysql> SELECT i, b FROM t_crypt;
+------+------------------+
| i    | data_crypt       |
+------+------------------+
| 1    | ?z#?uŠb?7?G[g,<  |
+------+------------------+

mysql> SELECT i, AES_DECRYPT(b,'crypt_key') AS data_crypt FROM t_crypt;
+------+----------------+
| i    | data_crypt     |
+------+----------------+
| 1    | Phrase cryptée |
+------+----------------+
```