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.

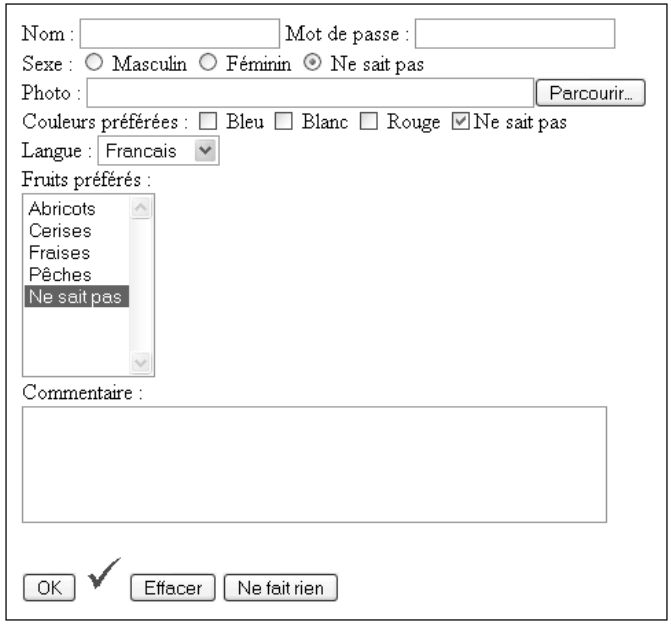
Entre les balises `<form>` et `</form>`, il est possible de placer des balises `<input>`, `<select>` ou `<textarea>` pour définir des zones de saisie.

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

MariaDB

Administration et optimisation

Collection
Ressources Informatiques

Extrait

Chapitre 4
Sécurité et gestion des utilisateurs

1. Introduction

La sécurité des données du système informatique de l'entreprise n'est pas seulement l'affaire du RSSI (Responsable de la sécurité des systèmes d'information). Les techniques et les bonnes pratiques nécessaires pour se protéger doivent être connues et appliquées, par tout un chacun. Les éléments de la chaîne applicative étant interdépendants, la robustesse de cette chaîne va dépendre de la solidité de son maillon le plus faible. Un test non effectué par l'application, un compte utilisateur qui a trop de droits sur la base de données, une mise à jour de sécurité non effectuée ou tout simplement une maladresse sur le serveur de production peuvent être à l'origine du vol ou de l'altération des numéros de cartes bancaires de vos clients par exemple. Les conséquences peuvent alors être désastreuses tant financièrement qu'en terme d'image pour la société. Certes la sécurité absolue n'existe pas, mais nous allons dans ce chapitre essayer de vous donner toutes les pratiques nécessaires pour minimiser les risques.

2. Sécurisation du serveur MariaDB

Les questions liées à la sécurité doivent être posées dès l'installation du serveur de base de données. MariaDB dérive étroitement de MySQL, or 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. L'inconvénient majeur est que son installation par défaut est très permissive, notamment en matière de sécurité...

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` :

```
$ 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/mariadb/
$ 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 Mettre un mot de passe au compte utilisateur root

C'est le super-utilisateur de MariaDB (à ne pas confondre avec l'administrateur système `root` sous UNIX) ; il a donc tous les droits sur le serveur MariaDB ; 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> SET PASSWORD FOR root@localhost = password('m0T2p4ss3'); /*
la commande set password permet de mettre un mot de passe à un compte
utilisateur */
```

Comme vous allez le voir un peu plus loin, un utilisateur MariaDB 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 MariaDB (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.

```
mysql> SELECT user, host, password FROM mysql.user WHERE user = 'root' \G
*************************** 1. row ***************************
user: root
host: localhost
password: *228F5395471FEFD9B0BB291954D2189384329691
*************************** 2. row ***************************
```

```
user: root
host: 123.456.78.9
password: *63D85DCA15EAFFC58C908FD2FAE50CCBC60C4EA2

mysql> DROP USER 'root'@'123.456.78.9' ;

mysql> SELECT user, host, password FROM mysql.user WHERE user = 'root' \G
*************************** 1. row ***************************
user: root
host: localhost
password: *228F5395471FEFD9B0BB291954D2189384329691
```

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

Dans le but de faciliter la prise en main du logiciel, un compte anonyme, c'est-à-dire un compte qui a le champ `user` à vide, peut être présent lors de l'installation de votre serveur. Il permet de se connecter au serveur avec n'importe quel utilisateur, en local et sans mot de passe. Il n'est guère besoin d'argumenter plus pour comprendre que, sur un serveur de production, un tel utilisateur n'est pas à sa place, il faut donc le supprimer. Ce principe doit être étendu à tous les comptes inutilisés, car c'est autant de problèmes potentiels en moins.

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

mysql> DROP USER ''@localhost;

mysql> SELECT user, host, password FROM mysql.user WHERE user = '' \G
```

2.1.4 Supprimer le schéma test

Le schéma `test` est créé par MariaDB lors de l'installation pour toutes les versions antérieures à 10.0 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.

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

mysql> DROP SCHEMA test;

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

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` suivi d'un underscore (_).

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.

```
$ mysql_secure_installation
NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL
MySQL SERVERS IN PRODUCTION USE! PLEASE READ EACH STEP CAREFULLY!
```

```
In order to log into MySQL to secure it, we'll need the current
password for the root user. If you've just installed MySQL, and
you haven't set the root password yet, the password will be blank,
so you should just press enter here.

Enter current password for root (enter for none):
```

2.2 Utilisation de SSL

Les données circulent en clair entre le client et le serveur MariaDB. Il est possible de les crypter en utilisant le protocole SSL (*Secure Sockets Layer*), qui permet d'assurer la sécurité, l'intégrité et la confidentialité des données échangées. Cependant, les traitements supplémentaires induits ont un impact non négligeable sur les performances, en les réduisant d'environ 30 %.

2.2.1 Les options

SSL propose une liste d'options. Plus d'informations sont disponibles sur le site de MariaDB : https://mariadb.com/kb/en/mariadb/secure-connections/. en voici une brève description :

- L'option `ssl` permet au serveur d'autoriser les connexions SSL et au client de se connecter en utilisant ce protocole.
- L'option `ssl-ca` permet de spécifier le fichier qui contient le certificat d'autorité (CA).
- L'option `ssl-capath` est le répertoire des fichiers contenant les certificats d'autorité SSL.
- L'option `ssl-cert` est le nom du certificat SSL à utiliser pour établir une connexion sécurisée.
- L'option `ssl-key` est le nom du fichier de la clé SSL à utiliser pour établir une connexion sécurisée.
- L'option `ssl-cipher` est la liste des chiffrements (*cipher*) autorisés à utiliser avec SSL.