PHP et MYSQL
Applications avancées

Traitement du courrier électronique

Walid BOUARIFI
Ecole Nationale des Sciences Appliquées de Marrakech
Université Cadi Ayyad

1- Introduction

Aujourd’hui, de plus en plus de sites offrent un ***webmail*** à leurs clients, c’est-à-dire une application web leur permettant de recevoir, d’organiser et d’envoyer des courriers *via* leur navigateur;

Pour qu’un utilisateur puisse lire son courrier, il faut disposer d’un moyen de le connecter à son serveur de courrier, qui est, généralement, une machine différente de celle du serveur web.

Il faut également pouvoir interagir avec la boîte aux lettres de l’utilisateur pour savoir les messages qu’il a reçus et les traiter individuellement.

Pour étudier de manière concrète les problèmes soulevés par l'échange de données via HTTP (et leurs solutions), nous allons étudier une application très simplifiée d'envoi de courrier électronique. Il s'agit d'un script unique, **Mail.php**, qui fonctionne en deux modes :

1. Si aucune donnée ne lui est soumise, le script affiche un formulaire de saisie d'un e-mail. L'utilisateur peut alors entrer les données du formulaire et les soumettre. Elles sont transmises, au même script, via HTTP.

2. Si des données sont soumises, le script récupère les données et doit:

- envoyer l'e-mail,
- stocker l'e-mail dans la base de données,
- l'afficher en HTML pour confirmer la prise en charge de l'envoi.

Le moyen le plus simple d’envoyer du courrier électronique avec PHP consiste à utiliser la fonction mail(). Si votre serveur de courrier est correctement configuré et que vous n’avez besoin de n’envoyer des messages qu’à vous-même, c’est sûrement la seule fonction dont vous aurez besoin.

Les utilisateurs doivent pouvoir lire, répondre, faire suivre et supprimer les messages de courriers existants, ainsi qu’en envoyer de nouveaux. Toutes les parties concernant la lecture peuvent s’effectuer avec IMAP ou POP3 et tout ce qui concerne l’envoi doit s’effectuer avec SMTP *via mail().*

Les deux protocoles principaux reconnus par les serveurs de courrier pour lire les boîtes aux lettres s’appellent POP3 (*Post Office Protocol version 3) et IMAP (Internet Message Access Protocol). Dans la mesure du possible, il est préférable de pouvoir* gérer les deux.

Voyons maintenant comment rassembler les pièces de ce puzzle.

Résumé de la solution

Ce qui nous intéresse ici, c'est le traitement des données transférées dans trois contextes différents : envoi sous forme de texte pur, insertion dans MySQL et affichage avec HTML. Chaque traitement est implanté par une fonction détaillée dans ce qui suit

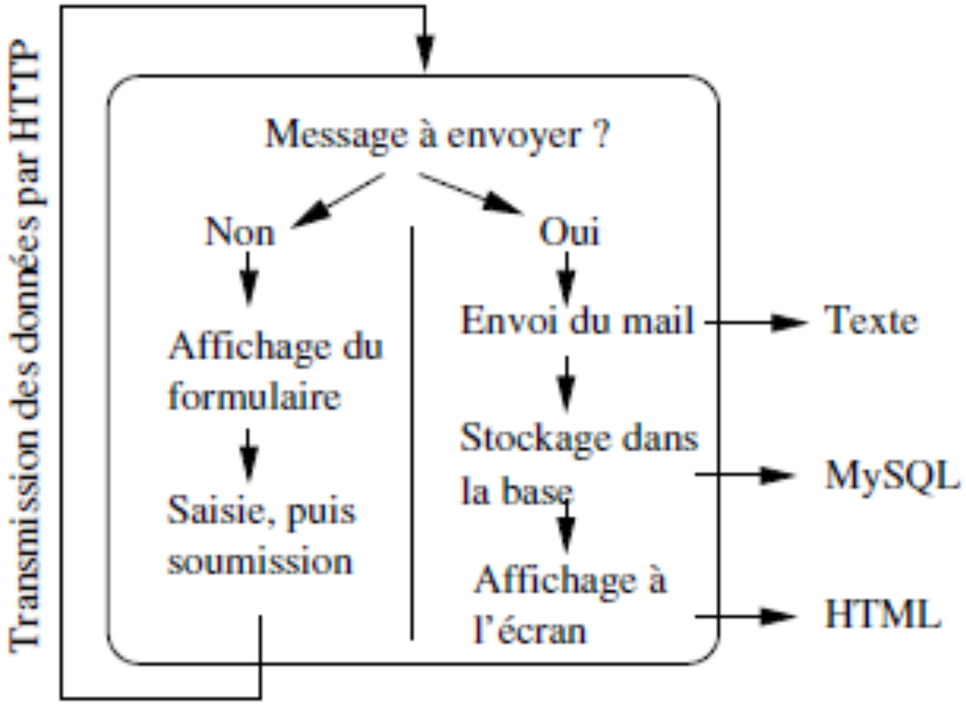

Figure 2.1 — Le schéma de l'application d'envoi d'un e-mail

1ère Etape : Créez une base Mails et une table Mail

CREATE TABLE Mails (id_mail INT AUTO_INCREMENT NOT NULL, destinataire VARCHAR(40) NOT NULL, sujet VARCHAR(40) NOT NULL, message TEXT NOT NULL, date_envoi DATETIME, PRIMARY KEY (id_mail));

INSERT INTO Mail (destinataire, sujet, message, date_envoi)

VALUES(’Walid.bouarifi@gmail.com’, ’Essai’, ’Test du mail’, NOW());

2ème Etape : Proposons un Script contenant les définitions de constantes pour la connexion à MySQL :

1- Fichier Connect.php

```
<?php
define ('NOM',"root");
define ('PASSE', "");
define ('SERVEUR', "localhost");
define ('BASE', "Mails"); ?>
```

2- Un script qui regroupe les fonctions pour la connexion à MYSQL, au serveur et à la base Mails: Connexion.php

3- Fichier pour exécuter une requête avec MYSQL fichier : ExecRequete.php

4- Fichier qui regroupe les fonctions et déclarations pour l'accès à MYSQL : Fichier UtilBD.php

3ème Etape: Comment obtenir du texte « pur » : envoi de l'e-mail?

Maintenant, l'e-mail sera envoyer grâce à la fonction PHP mail(). Les questions à se poser est le passage d'une représentation en texte brut à une représentation HTML. Si le texte à envoyer contient des mises en forme HTML, on peut les supprimer avec la fonction strip_tags(), comme le montre le script ci-dessous.

EnvoiMail.php

```php
<?php
function EnvoiMail ($mail){
$destinataire = $mail['destinataire']; // Extraction des paramètres
$sujet = $mail['sujet'];
// On retire toutes les balises HTML du message
$message = strip_tags($mail['message']);
// On va indiquer l'expéditeur, et placer nom@domaine.com en copie
$entete = "From: etudiant@domaine.com \r\n";
$entete .= "Cc: walid.bouarifi@gmail.com \r\n";
// Appel à la fonction PHP standard
mail ($destinataire, $sujet, $message, $entete);
} ?>
```

2. Le Script ci-dessous prend un tableau en paramètre, traite ses entrées en échappant les apostrophes, et exécute enfin la requête.

Fichier StockeMail.php

```
<?php
require_once ("UtilBD.php");
function StockeMail ($mail) {
$connexion = Connexion (NOM, PASSE, BASE, SERVEUR);
$destinataire = mysql_real_escape_string ($mail['destinataire']);
$sujet = mysql_real_escape_string($mail['sujet']);
$message = mysql_real_escape_string($mail['message']);
$requete = "INSERT INTO Mail(destinataire, sujet, message, date_envoi) " .
"VALUES ('$destinataire', '$sujet', '$message', NOW())";
ExecRequete ($requete, $connexion);}
?>
```

3. Script affichant un mail dans une page HTML.

Fichier AfficheMail.php

```php
<?php
function AfficheMail ($mail) {
$message = $mail['message'];
// On transforme les fins de lignes \n en <br/>
$message = nl2br($message);
// Et on affiche
echo "<b> l'email suivant a été envoyé avec succès: </b><br/>" .
$message;
} ?>
```

4ème Etape :

1. Script contrôlant l'entrée de l'application e-mail, le tableau en paramètre doit contenir les entrées: destinataire, sujet et message en vérifiant que les données ne sont pas vides.

Fichier ControleMail.php

```
<?php
function ControleMail (&$mail) {
if (!isSet($mail['destinataire'])) {
echo "Pas de destinataire!";
return false;}
else $mail['destinataire'] = htmlSpecialChars($mail['destinataire']);
....
// On vérifie que les données ne sont pas vides
if (empty($mail['destinataire'])){
echo "Destinataire vide!"; return false;
}
?>
```

2. Le script ci-après supprime tout échappement automatique des données HTTP dans un tableau de dimension quelconque, c'est-à-dire : echo n'effectue pas vraiment un affichage : il place le contenu demandé sur la sortie, c'est-à-dire dans le *code source* html.

Si votre contenu/texte contient des retours à la ligne, il seront bien insérés dans le code HTML mais le navigateur les ignorera (car en HTML, ils doivent être ignorés).

De même, si le contenu/texte contient des chevrons (surtout "<"), le navigateur pensera qu'il s'agit d'une balise HTML... c'est pourquoi il faut mettre < à la place.

Fichier **NormalisationHTTP.php**

```php
<?php
function NormalisationHTTP($tableau) {
// Parcours du tableau
foreach ($tableau as $cle => $valeur) {
if (!is_array($valeur)) // c'est un élément: on agit
$tableau[$cle] = stripSlashes($valeur);
else // c'est un tableau : on appelle récursivement
$tableau[$cle] = NormalisationHTTP($valeur);
}
return $tableau;
}
?>
```

3. Application de la suppression des échappements, si nécessaire, dans tous les tableaux contenant des données HTTP

Fichier Normalisation.php

```
<?php
require_once("NormalisationHTTP.php");
function Normalisation() {
// Si l'on est en échappement automatique, on rectifie...
if (get_magic_quotes_gpc()) {
$_POST = NormalisationHTTP($_POST);
$_GET = NormalisationHTTP($_GET);
$_REQUEST = NormalisationHTTP($_REQUEST);
$_COOKIE = NormalisationHTTP($_COOKIE);}
}
?>
```

5ème Etape : Formulaire basique pour l'envoi d'un e-mail.

6ème étape : Le script général, Mail.php, qui appelle ces différentes fonctions. Le script ***Mail.php*** *utilise une approche événementielle. Il sait quelle fonction appeler* pour chaque événement. Ici, les événements sont déclenchés lorsque l'utilisateur clique sur les différents boutons du site.

Fichier Mail.php

```
<?php  // Inclusion des fichiers contenant les déclarations de fonctions
require_once("ControleMail.php");
require_once("StockeMail.php");
require_once("AfficheMail.php");
require_once("EnvoiMail.php");
.....
StockeMail($_POST); // On a passé le test: stockage dans la base
AfficheMail ($_POST); // On affiche le texte de l'e-mail
EnvoiMail($_POST); } // Envoi de l'e-mail
else { // On affiche simplement le formulaire
require ("FormMail.html");
}
?>
```

L’étude de fonctionnalités plus avancées d’envoi d’e-mails (avec fichiers en attachement par exemple) peut être effectuée à l’aide des fonctions de php.net avec des fonctionnalités prêtes à l’emploi comme **phpMailer**.

PHPMailer est un paquetage Open Source de gestion du courrier électronique qui reconnaît les fichiers attachés, les destinataires multiples, l’authentification SMTP et qui dispose d’un grand nombre d’autres fonctionnalités. Il suffit d’inclure le fichier PHPMailer principal dans votre script et vous êtes prêt à envoyer du courrier.

Installation de PHPMailer: L’installation de PHPMailer s’effectue en suivant ces étapes :

1. Téléchargez les fichiers de PHPMailer à partir de **http://phpmailer.sourceforge.net/.**
2. Créez un répertoire *phpmailer sur votre serveur afin d’y stocker les fichiers* de PHPMailer.
3. Extrayez les fichiers de PHPMailer dans le répertoire *phpmailer.*
4. Choisissez votre méthode de transport du courrier.
5. Ajustez la configuration par défaut de PHPMailer en modifiant le contenu du fichier *class.phpmailer.php.*