Programmation Orienté Objet

PHP – Les mails

Envoyer un mail est une chose courante dans une application web.

Notez bien que cela fonctionne dans les 2 sens :

- l'internaute peut prendre contact avec le propriétaire d'un site à partir d'un formulaire de contact (ou d'inscription)
- le propriétaire d'un site envoie des mails aux internautes, par exemple :
  - envoi d'une confirmation d'inscription
  - envoi d'une facture en pièce jointe lors d'un achat en ligne
  - envoi d'une documentation, notice d'information, catalogue produits...
  - envoi d'une newsletter
  - rappel d'identifiant, réinitialisation d'un mot de passe

Pour cela, PHP propose la fonction `mail()`.

Attention : pour éviter le spam, la plupart des hébergeurs ne permettent pas l'utilisation de cette fonction (désactivée), ou en limitent l'usage (quota de mails envoyés sur une période donnée). Ils proposent parfois une alternative (arguments limités, surcharge par d'autres fonctions). Il faut en tenir compte dans les critères de choix d'un hébergeur.

Un mail est structuré en 2 parties :

- Une partie entête (métadonnées techniques) : *headers*.
- Une partie corps du message : *body*.
- On peut également vouloir envoyer un mail avec une pièce jointe.

L'envoi d'un mail ne relève pas que du langage informatique dans lequel il sera écrit (ici en PHP).

Plusieurs normes informatiques (RFC 2822, RFC 2047...), logiciels serveurs et protocoles de communication (SMTP etc.) entrent en jeu, ainsi que la configuration des serveurs d'envoi et de réception, des hébergeurs ou encore des fournisseurs de messagerie (Gmail, Yahoo, Orange, Free…), des logiciels de lecture clients (Outlook, Thunderbird...) et filtres antispam.

Autant de points qui rendent compliqué de s'assurer de la bonne réception d'un courriel par un destinataire.

Enfin pour faciliter l'utilisation de mails il existe :

- des librairies externes (*PHPMailer* par exemple)
- des outils internes dans les CMS
- des outils intégrés dans les frameworks (par exemple la librairies *Email* dans CodeIgniter ou *Swift* dans Symfony).

I.1. Configuration d'un serveur de mail

Par défaut, Wamp ne permet pas l'envoi de mail.

Il est nécessaire de :

- Installer un programme de serveur de mails (*sendmail*, *postfix* etc., issus du monde Linux)
- Configurer Wamp

I.2. Installation d'un serveur mail

mailhog permet d'émuler un serveur mail type Linux (*sendmail*, *postfix* etc., issus du monde Linux) dans un environnement Windows en local tel que Wamp.

- Dans *C:/wamp*, créez un répertoire nommé *mailhog*.
- Téléchargez le fichier *Windows (x64) client* à cette adresse
- Placez le fichier téléchargé *MailHog_windows_amd64.exe* dans votre répertoire *C:/wamp/mailhog/*
- Renommez le fichier *mailhog.exe*.

Attention : Mailhog n'est qu'un émulateur qui ne fait qu'intercepter et afficher les mails sur le SMTP local; c'est juste un outil de mise au point des mails en phase de développement et non pas un serveur de mails complet type *sendmail*.

I.3. Configuration de Wamp

- Ouvrez le fichier *php.ini* (fichier de configuration de PHP) : clic gauche sur l'icône Wamp > *PHP* > *php.ini*.
- Recherchez la section *[mail function]*
- Renseignez les valeurs suivantes :
  - `SMTP = 127.0.0.1`
  - `smtp_port = 1025`
  - `sendmail_path = "C:/wamp/mailhog/mailhog.exe sendmail"`
  - `mail.log = "c:/wamp/logs/mails.log"`
- Redémarrez Wamp.

Pour activer une variable dans *php.ini*, il faut éventuellement la décommenter en supprimant le `;` en début de ligne.

I.4. Lancement de mailhog

- Exécutez le fichier *C:/wamp/mailhog/mailhog.exe*
- Ouvrez l'adresse http://localhost:8025 dans votre navigateur

I.5. Coder l'envoi d'un mail

I.5.a. Utilisation de base

La syntaxe de la fonction PHP `mail()` est la suivante :

```
mail(destinataire, objet, message, entêtes facultatives, paramètres facultatifs)
```

Exemple (à remplacer l'adresse du destinataire par la vôtre) :

```
<?php
mail("monadresse@mail.com", "Confirmation d'inscription", "Bienvenue sur notre site");
```

La fonction `mail()` s'utilise donc avec plusieurs arguments :

- Destinataire : l'adresse mail de la personne à laquelle on veut envoyer le mail

- Objet : le sujet du mail, tel qu'il apparaîtra dans les clients de messagerie
- Message : le contenu principal du mail, tel qu'il apparaîtra dans les clients de messagerie
- Entêtes (headers) : des informations techniques additionnelles (facultatives).
- Paramètres : des informations techniques additionnelles (facultatives).

Ce qui donne :

```
<?php
$destinataire = "dave.loper@afpa.fr";
$objet = "Confirmation d'inscription" ;
$message = "Bienvenue sur notre site";

$entetes = array(
'From' => 'contact@jarditou.com',
'Reply-To' => 'commercial@jarditou.com',
'X-Mailer' => 'PHP/'.phpversion()
);

mail($destinataire, $objet, $message, $entete);
```

Notez bien que la fonction renvoie un booléen : `TRUE` si exécutée correctement, `FALSE` si échec.

I.5.b. Le(s) destinataire(s)

Le destinataire peut être écrit sous différentes formes.

Le destinataire peut être écrit sous différentes formes, mais le formatage de cette chaîne doit correspondre avec la norme RFC 2822.

Exemples les plus courants :

- `dave.loper@afpa.fr`
- `Dave Loper <dave.loper@afpa.fr>`

Il est possible d'indiquer plusieurs destinataires (attention, les filtres antispam n'aiment pas ça), dans ce cas les noms doivent être séparés par une virgule :

```
<?php
$destinataire = "Dave Loper <dave.loper@afpa.fr>,
jessica.pikatchien@laposte.net, alain.terieur@gmail.com";
```

I.5.c. L'objet

L'objet est le sujet du mail. Il doit lui aussi répondre à la norme RFC 2822.

Il doit respecter des bonnes pratiques de rédaction, sous peine d'être classé comme spam; cela relève plus du marketing.

I.5.d. Le corps du message

Le corps est donc la partie qui va comprendre le texte de votre message, c'est-à-dire la partie affichée, à lire.

Celui-ci peut-être au format texte ou HTML; il est possible de propsoer pour un même mail une configuration prenant en compte ces 2 alternatives.

I.5.e. Les entêtes

Les entêtes - *headers* - sont des informations techniques additionnelles, telles que :

- le type de présentation (texte ou HTML),
- l'encodage
- les destinataires en copie
- les pièces jointes

Ces informations sont facultatives mais dans la réalité, certaines sont exigées par les services de messageries (Gmail, Outlook, Yahoo!...) sinon le mail sera considéré comme spam. **Les entêtes ne sont pas visibles lorsque vous visualisez un mail : il faut afficher le code source du mail pour les voir**.

I.6. Envoyer une pièce jointe

https://a-pellegrini.developpez.com/tutoriels/php/mail/

I.7. Bonnes pratiques

- un balisage HTML correct
- définir l'encodage (UTF-8)
- un mail en HTML responsive pour qu'il puisse être lisible aussi bien sur PC que sur smartphones ou tablettes
- Eviter le spam : votre serveur (nom de domaine) pourrait être blacklisté comme spammeur pour longtemps (il existe des bases de données des spammeurs sur lesquelles se fondent les logiciels antispam). Vous pouvez cependant vous retrouvez avec un mail classé en spam suite à une mauvaise configuration du serveur d'envoi ou un contenu mal interprété (objet, textes etc.)

I.8. Ressources

- arobase.org
- Il existe des librairies facilitant le codage des mails : par exemple en PHP SwiftMailer ou PHPMailer.