

Envoyer un mail en PHP

Par [Adrien Pellegrini](#)  - [Josselin Willette](#) 

Date de publication : 1 juillet 2007

Envoyer un mail en PHP n'est pas une tâche bien difficile. Personnaliser son mail et comprendre le fonctionnement des différents entêtes est bien plus compliqué.

1 - Introduction.....	3
1.1 - Remerciements.....	3
1.2 - Préambule.....	3
2 - Les entêtes (headers).....	4
Bcc.....	4
Cc.....	4
Content-Description.....	4
Content-Type.....	4
Content-Transfer-Encoding.....	5
Date.....	5
From.....	5
MIME-Version.....	5
Priority.....	6
Reply-To.....	6
Sender.....	6
Subject.....	6
To.....	6
X-Confirm-Reading-To.....	6
X-Mailer.....	7
X-Priority.....	7
3 - Mise en pratique.....	8
3.1 - Mail de base.....	8
3.2 - Spécifier un expéditeur différent.....	8
3.3 - Mail format HTML.....	8
3.4 - Mail simple avec pièce jointe.....	9
3.5 - Mail complet (HTML + pièces jointes).....	10
4 - Créer un formulaire de contact simple.....	12
4.1 - Formulaire HTML.....	12
4.2 - Récupération des informations et envoi du mail.....	12
4.3 - Sécurité.....	13
4.4 - Vérification des champs de formulaire.....	14
4.5 - Récapitulatif.....	17
5 - Quelques liens.....	20

1 - Introduction

1.1 - Remerciements

Tous mes remerciements à **wichtounet**, **buchs** et **Yogui** pour leur relecture.

1.2 - Préambule

La fonction **mail()** que je vais utiliser pour envoyer des mails ne requiert aucune installation. Elle est présente automatiquement dans les versions de PHP.

Toutefois il vous faut configurer le serveur SMTP qui permet d'envoyer des mails. Pour le configurer vous devez trouver deux lignes dans le fichier *php.ini* de votre serveur.

Ces deux lignes sont :

```
SMTP      "localhost"  
smtp_port "25"
```

Une fois votre server SMTP configuré, vous êtes prêt pour lire la suite de cet article.

2 - Les entêtes (headers)

Les entêtes doivent, en théorie, être séparés par un CRLF (\r\n).

Toutefois, il se peut qu'il y ait des webmail comme Gmail avec qui le CRLF ne marche pas. Il devra être remplacé par un LF (\n).

Bcc

Bcc, pour "Blind carbon copy", sert à lister les destinataires comme *To* et *Cc* à la différence que chaque destinataire ne saura pas qui se trouve sur la liste des destinataires.

```
Bcc: adresse@developpez.com
```

Cc

Cc, pour "Carbon copy", sert à lister les destinataires comme *To* et *Bcc* à la différence que chaque destinataire saura qui se trouve sur la liste des destinataires.

```
Cc: adresse@developpez.com
```

Content-Description

Content-Description est utilisé pour donner une description d'une partie d'un mail. Par exemple pour une image.

```
Content-Description: Le plus grand lac du monde
```

Content-Type

Content-Type définit le type du contenu du message. Généralement composé d'un type/sous-type. Le type par défaut est "Content-type: text/plain; charset=us-ascii".

```
Content-Type: text/plain; charset="iso-8859-1"
```

```
Content-Type: multipart/mixed; boundary="frontier"
```

```
Content-Type: image/jpeg
```

text/plain :

Plain Text n'est utilisé que pour envoyer une séquence de caractères non formatés.

Plain Text peut être complété avec un *charset* pour définir le jeu de caractères. S'il n'est pas spécifié, celui par défaut sera utilisé (US-ASCII).

multipart :

Le type multipart requiert un paramètre appelé **boundary**. Ce paramètre est une sorte de limite entre les différentes parties d'un mail au format multipart. Il existe plusieurs sous-types que voici :

multipart/mixed :

Ce type est utilisé pour les messages comportant plusieurs parties indépendantes les unes des autres. Le type par défaut est "text/plain".

multipart/alternative :

Ce type est utilisé pour spécifier une version différente à une même partie de message. Le client mail choisira alors la meilleure version pour lui. La version préférée est généralement placée à la fin.

multipart/digest :


Ce type est identique au "multipart/mixed" mais a comme type par défaut "message/rfc822" qui est le plus compatible avec la norme  **RFC934**.

image :

Ce type est utilisé pour indiquer que le message contient une image.

Quelques sous-types : image/jpeg, image/gif, image/png. [*]

audio :

Ce type est utilisé pour indiquer que le message contient une donnée audio.

Quelques sous-types : audio/ac3, audio/mp4. [*]

video :

Ce type est utilisé pour indiquer que le message contient une donnée vidéo.

Quelques sous-types : video/mpeg. [*]

application :

Ce type est utilisé dans le cas où il n'y en a pas d'autre.

application/octet-stream :

Ce type sert à indiquer que le message contient des données au format binaire.

application/postscript :

Ce type indique un programme PostScript.

Content-Transfer-Encoding

Content-Transfer-Encoding spécifie le type d'encodage.

Encodages possibles : 7bit, 8bit, binary, quoted-printable, base64, ietf-token, x-token.

```
Content-Transfer-Encoding: base64
```

Date

Date spécifie une date pour le message.

Cette date devra être au format suivant :

"les 3 premiers caractères du jour (Sun - Sat)" "date (1-31)" "les 3 premiers caractères du mois (Jan - Dec)" "année (2006)" "heure format 24h (hh:mm:ss)" "fuseau horaire (GMT)"

```
Date: Sun, 17 Sep 2006 09:33:12 GMT
```

Code permettant d'avoir le bon format avec la fonction date()

```
echo 'Date: '.date('r');
```

From

From contient l'adresse mail de l'expéditeur.

```
From: "Moi" moi@developpez.com
```

MIME-Version

MIME-Version spécifie qu'on utilise le format MIME et sa version.

```
MIME-Version: 1.0
```

Priority

Priority spécifie la priorité du mail. Il en existe 3 : "normal", "urgent" et "non-urgent".

```
Priority: normal
```

Reply-To

Reply-to spécifie une adresse autre que celle de l'expéditeur pour que le destinataire renvoie sa réponse à celle-là et pas celle du From.

```
Reply-To: autre@developpez.com
```

Sender

Sender spécifie l'authentique auteur du mail. Sender sera utilisé par exemple si celui qui envoie le mail n'est pas celui qui l'a écrit.

```
Sender: auteur@developpez.com
```

Subject

Subject spécifie le "titre" du message.

```
Subject: Mail de confirmation
```

To

To spécifie le premier destinataire.

```
To: "Destinataire" destinataire@developpez.com
```

```
To: Destinataire <destinataire@developpez.com>
```

```
To: <destinataire@developpez.com>
```

```
To: destinataire@developpez.com
```

X-Confirm-Reading-To

X-Confirm-Reading-To envoie un mail pour spécifier si le message a été reçu ou lu. Les clients de messagerie demandent habituellement si l'accusé peut être envoyé.

```
X-Confirm-Reading-To: adresse@developpez.com
```

X-Mailer

X-Mailer spécifie le logiciel d'envoi du mail.


```
X-Mailer: PHP/5.2.2
```

X-Priority

X-Priority spécifie la priorité du mail.

```
X-Priority: 3
```

3 - Mise en pratique

 Normalement la norme spécifie qu'il faut séparer les entêtes par un CRLF (\r\n). Cependant, certaines boîtes mail utilisent seulement un LF (\n). Donc si vous ne recevez pas votre mail essayez avec un LF.

3.1 - Mail de base

Simple mail avec un peu de texte.

Inconvénient : si vous envoyez ce mail depuis votre site web, vous allez voir apparaître dans l'expéditeur quelque chose comme "CGI-Mailer <cgi-mailer@server.com>"

```
// To
$to = 'truc@server.com';

// Subject
$subject = 'Developpez.com - Test Mail';

// Message
$msg = 'Developpez.com - Message du mail ...';

// Function mail()
mail($to, $subject, $msg);
```

3.2 - Spécifier un expéditeur différent

Pour résoudre l'inconvénient précédent, il faut utiliser l'entête "From". Cet exemple montre aussi comment utiliser l'entête Bcc.

```
// To
$to = 'truc@server.com';

// Subject
$subject = 'Developpez.com - Test Mail';

// Message
$msg = 'Developpez.com - Message du mail ...';

// Headers
$headers = 'From: Adrien Pellegrini <mail@server.com>'. "\r\n";
$headers .= 'Bcc: Moi <moi@server.com>; lui <lui@server2.com>'. "\r\n";
$headers .= "\r\n";

// Function mail()
mail($to, $subject, $msg, $headers);
```

3.3 - Mail format HTML

La plupart des clients mail comme Gmail, Hotmail, Yahoo Mail, ont une réglementation assez stricte au niveau du contenu du message.

Par exemple, il est inutile d'utiliser les balises <head>, <body>, <link>, <style> car elles seront automatiquement supprimées dans la plupart des cas.

Donc il n'est pas question de définir une feuille de style externe. Pour cela il faudra utiliser l'attribut *style* des balises. Il faut savoir aussi que toutes les propriétés du CSS ne sont pas prises en compte.

Pour avoir un mail le plus compatible possible, il vous faudra donc copier simplement le contenu de la balise <body> en utilisant le CSS dans les balises.


```
// To
$to = 'truc@server.com';

// Subject
$subject = 'Developpez.com - Test Mail';

// Headers
$headers = 'Mime-Version: 1.0'."\r\n";
$headers .= 'Content-type: text/html; charset=utf-8'."\r\n";
$headers .= "\r\n";

// Message
$msg = '<strong>Developpez.com</strong> - Message du mail ...';

// Function mail()
mail($to, $subject, $msg, $headers);
```

3.4 - Mail simple avec pièce jointe

Pour envoyer un mail avec une pièce jointe, il faut utiliser le Content-Type: multipart/mixed.

Tous les types "multipart" requièrent un paramètre appelé **boundary**.

Ce paramètre est une sorte de limite entre les différentes parties d'un mail (ici, le texte du mail et la pièce jointe).

Le *boundary* commence par 2 tirets (--) et termine par un CRLF. Le *boundary* doit être compris entre 1 et 70 caractères. Les **2 tirets** au début **sont obligatoires** pour des raisons de compatibilité avec une ancienne norme qui est la

 **RFC934**.

Le boundary est généralement une chaîne aléatoire.

```
// To
$to = 'truc@server.com';

// Subject
$subject = 'Developpez.com - Test Mail';

// clé aléatoire de limite
$boundary = md5(uniqid(microtime(), TRUE));

// Headers
$headers = 'From: Adrien Pellegrini <mail@server.com>'."\r\n";
$headers .= 'Mime-Version: 1.0'."\r\n";
$headers .= 'Content-Type: multipart/mixed;boundary='.$boundary."\r\n";
$headers .= "\r\n";

// Message
$msg = 'This is a multipart/mixed message.'."\r\n\r\n";

// Texte
$msg .= '--'.$boundary."\r\n";
$msg .= 'Content-type:text/plain;charset=utf-8'."\r\n";
$msg .= 'Content-transfer-encoding:8bit'."\r\n";
$msg .= 'Un message avec une pièce jointe.'."\r\n";

// Pièce jointe
$file_name = 'image.jpg';
if (file_exists($file_name))
{
    $file_type = filetype($file_name);
    $file_size = filesize($file_name);

    $handle = fopen($file_name, 'r') or die('File '.$file_name.' can t be open');
    $content = fread($handle, $file_size);
    $content = chunk_split(base64_encode($content));
    $f = fclose($handle);

    $msg .= '--'.$boundary."\r\n";
    $msg .= 'Content-type:'.$file_type';name='.$file_name."\r\n";
    $msg .= 'Content-transfer-encoding:base64'."\r\n";
    $msg .= $content."\r\n";
}
```

3.5 - Mail complet (HTML + pièces jointes)

- 10 -

```
<li>La sécurité dans les expressions régulières en PHP,  
par <em>Guillaume Rossolini</em></li>  
</ul>  
</div>  
</div>'. "\r\n";  
  
// Pièce jointe 1  
$file_name = 'image.jpg';  
if (file_exists($file_name))  
{  
    $file_type = filetype($file_name);  
    $file_size = filesize($file_name);  
  
    $handle = fopen($file_name, 'r') or die('File '.$file_name.' can t be open');  
    $content = fread($handle, $file_size);  
    $content = chunk_split(base64_encode($content));  
    $f = fclose($handle);  
  
    $msg .= '--'.$boundary."\r\n";  
    $msg .= 'Content-type:'.$file_type.';name='.$file_name."\r\n";  
    $msg .= 'Content-transfer-encoding:base64'."\r\n\r\n";  
    $msg .= $content."\r\n";  
}  
  
// Pièce jointe 2  
$file_name = 'image2.jpg';  
if (file_exists($file_name))  
{  
    $file_type = filetype($file_name);  
    $file_size = filesize($file_name);  
  
    $handle = fopen($file_name, 'r') or die('File '.$file_name.' can t be open');  
    $content = fread($handle, $file_size);  
    $content = chunk_split(base64_encode($content));  
    $f = fclose($handle);  
  
    $msg .= '--'.$boundary."\r\n";  
    $msg .= 'Content-type:'.$file_type.';name='.$file_name."\r\n";  
    $msg .= 'Content-transfer-encoding:base64'."\r\n\r\n";  
    $msg .= $content."\r\n";  
}  
  
// Fin  
$msg .= '--'.$boundary."\r\n";  
  
// Function mail()  
mail($to, $subject, $msg, $headers);
```

4 - Créer un formulaire de contact simple

Vous avez envie d'intégrer un formulaire de contact sur votre site Web mais éviter d'utiliser mailto qui ouvre le logiciel de messagerie de vos visiteurs ?

Cette partie va vous montrer comment créer un formulaire de contact qui vous envoie un e-mail de façon transparente pour l'utilisateur.

4.1 - Formulaire HTML

Nous allons commencer par créer un formulaire HTML qui comprendra les champs :

- civilité
- nom/prénom
- adresse e-mail
- sujet
- message

Voici ce que ça donne :

```
<form action="send_email.php" method="post">
  <p>
    <label for="civilite">Civilité :</label>
    <select id="civilite" name="civilite">
      <option value="mr" selected="selected">Monsieur</option>
      <option value="mme">Madame</option>
      <option value="mlle">Mademoiselle</option>
    </select>
  </p>
  <p>
    <label for="nom">Nom/Prénom :</label>
    <input type="text" id="nom" name="nom" />
  </p>
  <p>
    <label for="email">E-mail :</label>
    <input type="text" id="email" name="email" />
  </p>
  <p>
    <label for="sujet">Sujet :</label>
    <input type="text" id="sujet" name="sujet" />
  </p>
  <p>
    <label for="message">Message :</label>
    <textarea id="message" name="message" cols="40" rows="4"></textarea>
  </p>
  <p>
    <input type="submit" name="envoye" value="Envoyer" />
  </p>
</form>
```

4.2 - Récupération des informations et envoi du mail

Le formulaire ci-dessus envoie les informations en POST, donc nous les récupérons grâce à la variable PHP `$_POST` dans notre fichier `send_email.php`.

```
/* Récupération des valeurs des champs du formulaire */
if (get_magic_quotes_gpc())
{
    $civilite = stripslashes($_POST['civilite']);
    $nom = stripslashes($_POST['nom']);
    $expediteur = stripslashes($_POST['email']);
}
```

```
$sujet = stripslashes($_POST['sujet']);
$message = stripslashes($_POST['message']);
}
else
{
    $civillite = $_POST['civillite'];
    $nom = $_POST['nom'];
    $expediteur = $_POST['email'];
    $sujet = $_POST['sujet'];
    $message = $_POST['message'];
}
```

Maintenant, passons à la phase de création de l'e-mail :

```
/* Destinataire (votre adresse e-mail) */
$to = 'moi@domaine.com';

/* Construction du message */
$msg = 'Bonjour, ' . "\r\n\r\n";
$msg .= 'Ce mail a été envoyé depuis monsite.com par ' . $civillite . ' ' . $nom . "\r\n\r\n";
$msg .= 'Voici le message qui vous est adressé : ' . "\r\n";
$msg .= '*****' . "\r\n";
$msg .= $message . "\r\n";
$msg .= '*****' . "\r\n";

/* En-têtes de l'e-mail */
$headers = 'From: ' . $nom . ' <' . $expediteur . '>' . "\r\n\r\n";

/* Envoi de l'e-mail */
mail($to, $sujet, $msg, $headers);
```

Et voilà, e-mail envoyé !

4.3 - Sécurité

Notre formulaire de contact étant dès lors fonctionnel, il est vulnérable à différents types d'"attaques" dont il faut impérativement se protéger :

- les injections de headers
- le spam d'e-mails indésirables envoyés par des robots
- les insertions de scripts via le formulaire (uniquement dans le cas d'e-mails HTML)

Pour nous immuniser des injections de headers, nous devons écrire une expression régulière qui va vérifier si le contenu des champs n'en contient pas :

```
/* Expression régulière permettant de vérifier qu'aucun en-tête n'est inséré dans nos champs */
$regex_head = '/[\n\r]/';

/* On vérifie qu'il n'y a aucun header dans les champs */
if (preg_match($regex_head, $expediteur)
    || preg_match($regex_head, $nom)
    || preg_match($regex_head, $sujet))
{
    $alert = 'En-têtes interdites dans les champs du formulaire';
}
else
{
    /* envoi de l'e-mail */
}

/* On affiche l'erreur s'il y en a une */
if (!empty($alert))
{
    echo $alert;
}
```

Aussi, il faut nous défendre des robots spammeurs. Pour cela, nous allons vérifier la page appelant notre script d'envoi (referer). Admettons que la page sur laquelle est notre script soit la même que celle qui contient le formulaire. Par exemple, notre page formulaire s'appelle `send_email.php` et la valeur de l'attribut action de la balise form est aussi `send_email.php`. La page se rappelle donc elle-même et son referer est alors elle-même également. Nous testons donc si le referer est bien notre page :

```
/* Si le formulaire n'est pas posté de notre site on renvoie vers la page d'accueil */
if ($_SERVER['HTTP_REFERER'] != 'http://www.monsite.com/send_email.php')
{
    header('Location: http://www.monsite.com/');
}
else
{
    /* envoi de l'e-mail */
}
```

Enfin, pour éviter les insertions de scripts dans le cas d'un e-mail HTML, nous pouvons protéger nos variables `$_POST` de cette manière :

```
/* Récupération des valeurs des champs du formulaire */
if (get_magic_quotes_gpc())
{
    $civillite = stripslashes(htmlentities($_POST['civillite']));
    $nom = stripslashes(htmlentities($_POST['nom']));
    $expediteur = stripslashes(htmlentities($_POST['email']));
    $sujet = stripslashes(htmlentities($_POST['sujet']));
    $message = stripslashes(htmlentities($_POST['message']));
}
else
{
    $civillite = htmlentities($_POST['civillite']);
    $nom = htmlentities($_POST['nom']);
    $expediteur = htmlentities($_POST['email']);
    $sujet = htmlentities($_POST['sujet']);
    $message = htmlentities($_POST['message']);
}
```

L'utilisation de **htmlentities()** va convertir les caractères spéciaux tels que le signe inférieur et supérieur en entités HTML (< remplacé par <) et donc interdire l'exécution de scripts.

4.4 - Vérification des champs de formulaire

Vous n'avez sans doute pas envie de recevoir des e-mails vides, nous devons donc vérifier avant de les envoyer si aucun des champs n'est vide.

Nous ajoutons donc cette vérification à notre script :

```
if (empty($civillite)
    || empty($nom)
    || empty($expediteur)
    || empty($sujet)
    || empty($message))
{
    $alert = 'Tous les champs doivent être renseignés';
}
else
{
    /* envoi de l'e-mail */
}

/* On affiche l'erreur s'il y en a une */
if (!empty($alert))
{
```

```
    echo $alert;
}
```

Oui, mais si l'utilisateur ne met que des espaces ? Dans ce cas-là il faut modifier la récupération des variables comme suit :

```
/* Récupération des valeurs des champs du formulaire */
if (get_magic_quotes_gpc())
{
    $civilite = stripslashes(trim($_POST['civilite']));
    $nom = stripslashes(trim($_POST['nom']));
    $expediteur = stripslashes(trim($_POST['email']));
    $sujet = stripslashes(trim($_POST['sujet']));
    $message = stripslashes(trim($_POST['message']));
}
else
{
    $civilite = trim($_POST['civilite']);
    $nom = trim($_POST['nom']);
    $expediteur = trim($_POST['email']);
    $sujet = trim($_POST['sujet']);
    $message = trim($_POST['message']);
}
```

La fonction **trim()** supprime tous les espaces en début et fin de chaîne.

Un petit check de plus ne pouvant pas faire de mal, nous allons contrôler que le format de l'adresse e-mail saisie est correct :

```
/* Expression régulière permettant de vérifier si le format d'une adresse e-mail est correct */
$regex_mail = '/^[+-.\\w]{1,64}@[-.\\w]{1,64}\\.[-\\.\\w]{2,6}$/i';

/* On vérifie que le format de l'e-mail est correct */
if (!preg_match($regex_mail, $expediteur))
{
    $alert = 'L adresse '.$expediteur.' n est pas valide';
}
else
{
    /* envoi de l'e-mail */
}
```

Une dernière vérification nous permettra d'éviter qu'une personne n'envoie deux fois l'e-mail en rafraichissant sa page. Pour ce faire, un cookie de courte durée fera très bien l'affaire :

```
if (!isset($_COOKIE['sent']))
{
    /* envoi de l'e-mail */

    /* [...] */

    if (mail($to, $sujet, $msg, $headers))
    {
        /* On crée un cookie de courte durée (ici 120 secondes) pour éviter de
         * renvoyer un e-mail en rafraichissant la page */
        setcookie('sent', '1', time() + 120);
    }
}
```

Pour finir, un petit plus qui va augmenter l'ergonomie de notre formulaire : lorsque l'utilisateur voit afficher un message d'erreur, avec le code actuel ses champs se vident et il perd donc toutes ses informations, ce qui peut être assez gênant lorsqu'il a écrit un long message et doit le resaisir.

Il faut donc modifier notre formulaire pour y afficher les valeurs que l'utilisateur a entré et les détruire dans le cas où l'e-mail a bien été envoyé :

```
<form action="index.php" method="post">
  <p>
    <label for="civilite">Civilité :</label>
    <select id="civilite" name="civilite">
      <option
        value="mr"
        <?php
          if (!isset($_POST['civilite']) || $_POST['civilite'] == 'mr')
          {
            echo ' selected="selected"';
          }
        ?>
      >
        Monsieur
      </option>
      <option
        value="mme"
        <?php
          if (isset($_POST['civilite']) && $_POST['civilite'] == 'mme')
          {
            echo ' selected="selected"';
          }
        ?>
      >
        Madame
      </option>
      <option
        value="mlle"
        <?php
          if (isset($_POST['civilite']) && $_POST['civilite'] == 'mlle')
          {
            echo ' selected="selected"';
          }
        ?>
      >
        Mademoiselle
      </option>
    </select>
  </p>
  <p>
    <label for="nom">Nom/Prénom :</label>
    <input type="text" id="nom" name="nom"
      value="<?php echo (isset($_POST['nom'])) ? $nom : ' ' ?>"
    />
  </p>
  <p>
    <label for="email">E-mail :</label>
    <input type="text" id="email" name="email"
      value="<?php echo (isset($_POST['email'])) ? $expediteur : ' ' ?>"
    />
  </p>
  <p>
    <label for="sujet">Sujet :</label>
    <input type="text" id="sujet" name="sujet"
      value="<?php echo (isset($_POST['sujet'])) ? $sujet : ' ' ?>"
    />
  </p>
  <p>
    <label for="message">Message :</label>
    <textarea id="message" name="message" cols="40" rows="4">
    <?php echo (isset($_POST['message'])) ? $message : ' ' ?>
    </textarea>
  </p>
  <p>
    <input type="submit" name="envoyer" value="Envoyer" />
  </p>
</form>
```

```
if (mail($to, $sujet, $msg, $headers))
```



```
{  
    /* On détruit la variable $_POST */  
    unset($_POST);  
}
```

4.5 - Récapitulatif

Dans la partie précédente, nous avons vu différents morceaux de code essayés, nous allons regrouper le tout pour faire un formulaire complet.

Voici au final, notre page `send_email.php` :

```
<?php  
/* Si le formulaire est envoyé alors on fait les traitements */  
if (isset($_POST['envoyer']))  
{  
    /* Récupération des valeurs des champs du formulaire */  
    if (get_magic_quotes_gpc())  
    {  
        $civillite = stripslashes(trim($_POST['civillite']));  
        $nom       = stripslashes(trim($_POST['nom']));  
        $expediteur = stripslashes(trim($_POST['email']));  
        $sujet     = stripslashes(trim($_POST['sujet']));  
        $message   = stripslashes(trim($_POST['message']));  
    }  
    else  
    {  
        $civillite = trim($_POST['civillite']);  
        $nom       = trim($_POST['nom']);  
        $expediteur = trim($_POST['email']);  
        $sujet     = trim($_POST['sujet']);  
        $message   = trim($_POST['message']);  
    }  
  
    /* Expression régulière permettant de vérifier si le  
    * format d'une adresse e-mail est correct */  
    $regex_mail = '/^[+-.\\w]{1,64}@[-.\\w]{1,64}\\.[-.\\w]{2,6}$/i';  
  
    /* Expression régulière permettant de vérifier qu'aucun  
    * en-tête n'est inséré dans nos champs */  
    $regex_head = '/[\\n\\r]/';  
  
    /* Si le formulaire n'est pas posté de notre site on renvoie  
    * vers la page d'accueil */  
    if($_SERVER['HTTP_REFERER'] != 'http://www.monsite.com/send_email.php')  
    {  
        header('Location: http://www.monsite.com/');  
    }  
    /* On vérifie que tous les champs sont remplis */  
    elseif (empty($civillite)  
            || empty($nom)  
            || empty($expediteur)  
            || empty($sujet)  
            || empty($message))  
    {  
        $alert = 'Tous les champs doivent être renseignés';  
    }  
    /* On vérifie que le format de l'e-mail est correct */  
    elseif (!preg_match($regex_mail, $expediteur))  
    {  
        $alert = 'L\'adresse '.$expediteur.' n\'est pas valide';  
    }  
    /* On vérifie qu'il n'y a aucun header dans les champs */  
    elseif (preg_match($regex_head, $expediteur)  
            || preg_match($regex_head, $nom)  
            || preg_match($regex_head, $sujet))  
    {  
        $alert = 'En-têtes interdites dans les champs du formulaire';  
    }  
}
```

```

/* Si aucun problème et aucun cookie créé, on construit le message et on envoie l'e-mail */
elseif (!isset($_COOKIE['sent']))
{
    /* Destinataire (votre adresse e-mail) */
    $to = 'moi@domaine.com';

    /* Construction du message */
    $msg = 'Bonjour, '.$_POST['civilite'].' '.$nom."\n\n";
    $msg .= 'Ce mail a été envoyé depuis monsite.com par '.$civilite.' '.$nom."\n\n";
    $msg .= 'Voici le message qui vous est adressé : '.$message."\n\n";
    $msg .= '*****\n\n';
    $msg .= $message."\n\n";
    $msg .= '*****\n\n';

    /* En-têtes de l'e-mail */
    $headers = 'From: '.$nom.' <'.$expediteur.'>.\n\n';

    /* Envoi de l'e-mail */
    if (mail($to, $sujet, $msg, $headers))
    {
        $alert = 'E-mail envoyé avec succès';

        /* On crée un cookie de courte durée (ici 120 secondes) pour éviter de
        * renvoyer un mail en rafraichissant la page */
        setcookie("sent", "1", time() + 120);

        /* On détruit la variable $_POST */
        unset($_POST);
    }
    else
    {
        $alert = 'Erreur d\'envoi de l\'e-mail';
    }
}

/* Cas où le cookie est créé et que la page est rafraichie, on détruit la variable $_POST */
else
{
    unset($_POST);
}
}

?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html lang="fr">
<head>
    <title>Contactez moi</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
</head>
<body>

<?php
if (!empty($alert))
{
    echo '<p style="color:red">'.$alert.'</p>';
}
?>

<form action="send_email.php" method="post">
    <p>
        <label for="civilite">Civilité :</label>
        <select id="civilite" name="civilite">
            <option
                value="mr"
                <?php
                    if (!isset($_POST['civilite']) || $_POST['civilite'] == 'mr')
                    {
                        echo ' selected="selected"';
                    }
                ?>
            >
                Monsieur

```







```

        </option>
        <option
            value="mme"
            <?php
                if (isset($_POST['civilite']) && $_POST['civilite'] == 'mme')
                {
                    echo ' selected="selected"';
                }
            ?>
        >
            Madame
        </option>
        <option
            value="mlle"
            <?php
                if (isset($_POST['civilite']) && $_POST['civilite'] == 'mlle')
                {
                    echo ' selected="selected"';
                }
            ?>
        >
            Mademoiselle
        </option>
    </select>
</p>
<p>
    <label for="nom">Nom/Prénom :</label>
    <input type="text" id="nom" name="nom"
        value="<?php echo (isset($_POST['nom'])) ? $nom : ' ' ?>"
    />
</p>
<p>
    <label for="email">E-mail :</label>
    <input type="text" id="email" name="email"
        value="<?php echo (isset($_POST['email'])) ? $expediteur : ' ' ?>"
    />
</p>
<p>
    <label for="sujet">Sujet :</label>
    <input type="text" id="sujet" name="sujet"
        value="<?php echo (isset($_POST['sujet'])) ? $sujet : ' ' ?>"
    />
</p>
<p>
    <label for="message">Message :</label>
    <textarea id="message" name="message" cols="40" rows="4">
    <?php echo (isset($_POST['message'])) ? $message : ' ' ?>
    </textarea>
</p>
<p>
    <input type="submit" name="envoyer" value="Envoyer" />
</p>
</form>

</body>
</html>

```

5 - Quelques liens

-  **Compatibilité des propriétés CSS dans les emails**
-  **Envoyer des e-mails simplement en PHP grâce à PHP Mailer** (Stéphane Eyskens)
-  **Validation d'adresse e-mail en PHP** (Alexandre Tranchant)
-  **Le protocole SMTP** (Benjamin Roux)
-  **Types mime** (Cédric Chatelain)
-  **Norme RFC822**