


Php

Créer un formulaire de contact

Auteur	Aurélian Bellou-Bousselaire
Licence	
Version	2017-04-03

Pour créer un formulaire de contact, il faut réaliser deux parties :

- Une page de formulaire
- Une page php d'envoi (cette page sera celle dont le contenu HTML s'affichera après l'envoi)

Page de formulaire

On va réaliser une page de formulaire « classique » html :

```
<form id="contact" method="post" action="contact.php">
  <fieldset><legend>Vos coordonnées</legend>
    <p><label for="nom">Nom :</label><input type="text" id="nom" name="nom" tabindex="1" /></p>
    <p><label for="email">Email :</label><input type="text" id="email" name="email" tabindex="2" /></p>
  </fieldset>

  <fieldset><legend>Votre message :</legend>
    <p><label for="objet">Objet :</label><input type="text" id="objet" name="objet" tabindex="3" /></p>
    <p><label for="message">Message :</label><textarea id="message" name="message" tabindex="4" cols="30" rows="8"></textarea></p>
  </fieldset>

  <div style="text-align:center;"><input type="submit" name="envoi" value="Envoyer le formulaire !" /></div>
</form>
```

Cette page n'est qu'un exemple très simple qu'il est possible d'améliorer mais le plus important est qu'elle utilise la méthode « post » pour l'envoi et qu'elle renvoie vers « contact.php »

Page de traitement

C'est la page de traitement qui va réaliser l'ensemble des opérations nécessaires pour transformer le contenu du formulaire en envoi mail définitif. Il y a plusieurs opérations indispensables à réaliser :

- Vérifier que l'ensemble des champs obligatoires sont bien remplis et que leur contenu est conforme
- Vérifier que le formulaire a bien été envoyé via la page html et non pas en accès direct
- Transformer les informations récupérées en une version lisible dans un mail
- Enfin envoyer le dit mail.

Configuration

Nous débutons notre page par une configuration. Ces variables sont inscrites en début de page, ce qui permettra de les modifier facilement si besoin :

```
<?php
// Configuration
// copie ? (envoie une copie au visiteur)
$copie = 'non'; // 'oui' ou 'non'

// Destinataire
$destinataire = 'contact@sylvan-larochelle.com';
$message_envoye = "Votre message a bien été envoyé, l'équipe de Digital-Sup vous répondra dans les plus brefs délais. ";
$message_non_envoye = "L'envoi de votre message a échoué, merci de réessayer.";
$message_erreur = "Vous devez envoyer le formulaire via la page index";
$message_formulaire_invalide = "Certains champs sont invalides, merci de vérifier votre adresse e-mail.";
```

Cette fonction va permettre de nettoyer le texte de message pour le transformer en un texte lisible.

```

/*
 * cette fonction sert à nettoyer et enregistrer un texte
 */
function Rec($text){
    $text = htmlspecialchars(trim($text), ENT_QUOTES);
    if (1 === get_magic_quotes_gpc())
    {
        $text = stripslashes($text);
    }
    $text = nl2br($text);
    return($text);
};

```

Fonction htmlspecialchars : Certains caractères ont des significations spéciales en HTML, et doivent être remplacés par des entités HTML pour conserver leurs significations. Cette fonction retourne une chaîne de caractères avec ces modifications. Cette fonction php est intégrée par défaut dans les interpréteurs / serveurs php.

Get_magic_quotes_gpc : Cette fonction vérifie que la fonction du même nom est activée sur le serveur php. Si c'est le cas, on va utiliser la fonction stripslashes qui va échapper la variable \$text débarrassée des « magic quotes ». Sur les serveurs récents, il est maintenant inutile d'utiliser cette fonction mais l'avantage de laisser le if présent est que notre morceau de code fonctionne quelle que soit la version de php.

Nl2br : Cette fonction insère '
' devant chaque nouvelle ligne. Ceci permet de conserver les mises à la ligne dans le mail créé.

Fonction de vérification de l'adresse mail

```
* Cette fonction sert à vérifier la syntaxe d'un email
*/
function IsEmail($email) {

    $value = preg_match('/^(?:(?:[w\!\/#\$\%\&\'\"*\+|-|_|=|\^`\\{|\}~]+\.)*[w\!\/#\$\%\&\'\"*\+|-|_|=|\^`\\{|\}~]+@(?:(?:(?:[a-zA-Z
```

Cette fonction va vérifier que la syntaxe mail est bien respectée.

Si le mail est vide ou que la fonction `preg_match` (qui vérifie que le mail matche bien avec le masque indiqué derrière) renvoie `false`, la fonction renverra un `false` sinon elle renverra un `true`.

La syntaxe (test) ? valeur_si_oui : valeur_si_non ; est la version « allégée » de l'instruction « if » à n'utiliser que pour des test très simples comme celui-ci.

Récupération et vérification des variables

```
if (!isset($_POST['envoi']))
{
    $affichage = "<p>".$message_erreur."</p>";
}
else
{
    // formulaire envoyé, on récupère tous les champs.
    $nom      = (isset($_POST['nom']))      ? Rec($_POST['nom'])      : '';
    $email     = (isset($_POST['email']))    ? Rec($_POST['email'])    : '';
    $objet     = (isset($_POST['objet']))    ? Rec($_POST['objet'])    : '';
    $message   = (isset($_POST['message'])) ? Rec($_POST['message']) : '';
    $telephone = (isset($_POST['telephone'])) ? Rec($_POST['telephone']) : '';

    // On va vérifier les variables et l'email ...

    $email = (IsEmail($email)) ? $email : '';
    $objet = "Message envoyé depuis www.digital-sup.com : " . $objet;
    // var_dump ($email);
    // var_dump($message);
    // var_dump($telephone);
    // var_dump($objet);
}
```

L'ensemble des fonctions sont encapsulées dans un else.

Si le bouton d'envoi du formulaire n'a pas été utilisé et n'a pas renvoyé de contenu dans la variable « envoi », l'affichage va renvoyer un message d'erreur et la page s'arrête.

Sinon l'exécution continue. Cette syntaxe permet d'améliorer la sécurité du code.

On récupère l'ensemble des variables en vérifiant préalablement qu'elles existent. Les « if » simplifiés permettent de renvoyer le contenu de la variable si elle en a un, sinon ils renvoient du vide.

Concernant l'adresse mail, on la fait passer par la fonction de vérification puis dans un if simplifié. Si l'adresse mail est correcte, elle est encapsulée dans la variable \$email. Sinon cette dernière sera affublée d'un simple « ».

Les var_dump en bas de ce bloc permettent de vérifier le contenu des variables en cas de problème lors de la réalisation.

Vérification avant d'envoi du mail

```
if (($nom != '') && ($email != '') && ($objet != '') && ($message != ''))
{
    // les 4 variables sont remplies, on génère puis envoie le mail
    $headers = 'MIME-Version: 1.0' . "\r\n";
    $headers .= 'From: '.$nom.' <'.$email.'>' . "\r\n" .
        'Reply-To: '.$email. "\r\n" .
        'Content-Type: text/plain; charset="utf-8"; DelSp="Yes"; format=flowed' . "\r\n" .
        'Content-Disposition: inline' . "\r\n" .
        'Content-Transfer-Encoding: 7bit' . "\r\n" .
        'X-Mailer: PHP/' . phpversion();

    // envoyer une copie au visiteur ?
    if ($copie == 'oui')
    {
        $cible = $destinataire.';'.$email;
    }
    else
    {
        $cible = $destinataire;
    }
};
```

Grâce au if à conditions multiples, on vérifie qu'il y a bien quelque chose dans chaque variable et non juste un « » qui indiquerait une erreur.

Ensuite on génère le mail.

```
$message = "Nom : ".$nom . " Téléphone : " . $telephone . "\r\n" .
    " Objet: " . $objet. "\r\n".
    " Message : " . $message;

// Remplacement de certains caractères spéciaux
$message = str_replace("&#039;", "'", $message);
$message = str_replace("&#8217;", "'", $message);
$message = str_replace("&quot;", "'", $message);
$message = str_replace('<br>', '', $message);
$message = str_replace('<br />', '', $message);
$message = str_replace("&lt;", "<", $message);
$message = str_replace("&gt;", ">", $message);
$message = str_replace("&amp;", "&", $message);
```

Ne pas oublier de remplacer les caractères spéciaux par les véritables caractères et non leur code.

```
// Envoi du mail
$num_emails = 0;
$tmp = explode(';', $cible);
foreach($tmp as $email_destinataire)
{
    if (mail($email_destinataire, $objet, $message, $headers))
        $num_emails++;
}

if ((($copie == 'oui') && ($num_emails == 2)) || (($copie == 'non') && ($num_emails == 1)))
{
    $affichage = '<p>'.$message_envoye.'</p>';
}
else
{
    $affichage = '<p>'.$message_non_envoye.'</p>';
};
```

L'envoi du mail s'effectue dans un if qui va compter le nombre d'emails envoyés. Si ce nombre n'est pas conforme, un message indique que le message n'a pas été envoyé.

```
else
{
    // une des 3 variables (ou plus) est vide ...
    $affichage = '<p>'.$message_formulaire_invalide.' <a href="index.html#tf-contact">Retour au formulaire</a></p>'. "\n";
};
}; // fin du if (isset($_POST['envoi']))
?>
```

Ce else est celui du test des variables. Si l'une d'entre elles est vide, on renvoie le visiteur vers le formulaire.

La page HTML de confirmation

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Document sans titre</title>
</head>
<?php
echo $affichage;
?>
<body>
</body>
</html>
```

Enfin en bas de page, on crée une page HTML confirmant le bon envoi (ou non) du mail.

Bien sûr dans cet exemple, la page est très simple mais il est nécessaire de l'agréementer pour qu'elle respecte la charge graphique du site.

