

# Le langage PHP et les sites web

Dans le module SI4 vous avez découvert le langage PHP et réalisé parmi vos premiers programmes avec celui-ci. Formellement vous n'irez pas plus loin dans ce qui est des structures de programmation dans ce chapitre. Nous en verrons l'un des aspects essentiel, qui fait que PHP est aujourd'hui un langage parmi les plus utilisés : l'interaction avec les serveurs web. Le langage PHP, contrairement à Javascript, n'est pas exécuté du côté client. C'est un langage qui sert, du côté serveur, à générer des documents web dynamiquement en interaction avec une base de données.

## Sommaire

A. Le langage de PHP et le serveur web .....	2
A.1. Introduction .....	2
A.1.1. Notion de protocole .....	2
A.1.2. Transactions web .....	2
A.1.3. Interactions web .....	4
A.2. Éléments de base du langage PHP .....	5
A.2.1. Structure et forme de base d'un script PHP .....	5
A.2.2. Structures de programmation .....	7
A.2.3. Mise en page avec PHP .....	10
A.2.1. Utilisation de fonctions et de procédures .....	12
B. La transmission des données .....	16
B.1. Transmission des données passées par l'URL .....	16
B.1.1. Principe .....	16
B.1.2. Passage de paramètre simple .....	16
B.1.3. Le contrôle des paramètres .....	18
B.2. Transmission des données passées par formulaire .....	20
B.2.1. Transmission des données avec la méthode GET .....	20
B.2.2. Transmission des données avec la méthode POST .....	22
B.2.3. Page qui s'appelle elle-même .....	26
B.2.4. Sécurisation des données transmises .....	28
B.3. La transmission des fichiers .....	31

## A. Le langage de PHP et le serveur web

Le langage PHP a été conçu durant l'automne 1994 par Rasmus Lerdorf. Les premières versions (qui restèrent privées) étaient utilisées afin de savoir qui venait consulter son CV en ligne. La première version publique fut disponible au début de l'année 1995. PHP grandit de manière spectaculaire et de nombreuses personnes contribuèrent à son amélioration.

En 2007, 34% des entreprises utilisaient PHP (*selon le JDN*).

### A.1. Introduction

L'objectif de cette première partie est de comprendre le fonctionnement du web en général et le mécanisme des services web en particulier.

Cela implique de nous attarder sur les notions de :

- protocole,
- transactions web,
- interactions Apache.

#### A.1.1. Notion de protocole

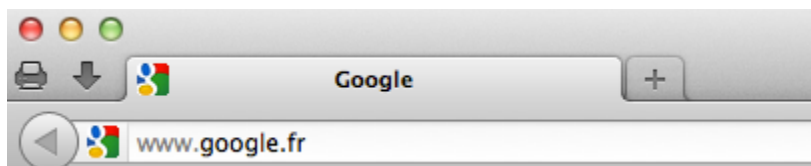
Un **protocole** est la façon selon laquelle les différents processus discutent entre eux. Pour assurer cette discussion il est nécessaire d'avoir des règles et des procédures uniformes. Ces règles et ces procédures sont définies par des normes internationales, respectées par tous et sans lesquelles aucune liaison ne serait durable ou même possible entre les réseaux.

Par exemple : Pour visiter un site web le protocole HTTP (*HyperText Transfer Protocol*) est utilisé par le serveur et le navigateur pour communiquer. S'il s'avérait que l'un ou l'autre n'utilisait pas les protocoles dédiés aux sites web, il serait impossible de visiter ce site.

#### A.1.2. Transactions web

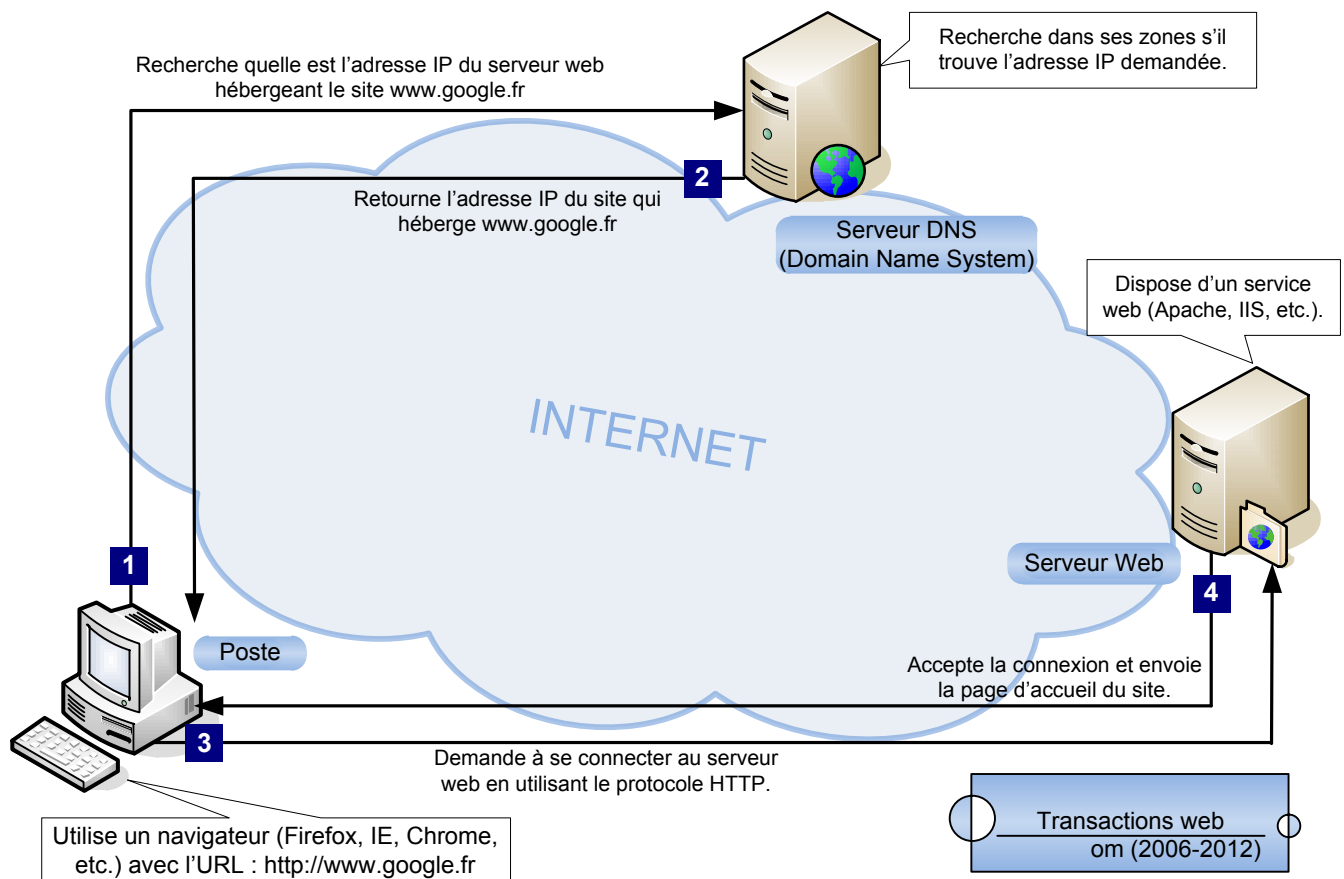
Lorsqu'un internaute saisit une adresse réticulaire dans la zone URL (*Uniform Resource Locator*) il déclenche une suite de transactions qui vont lui permettre de visiter le site demandé.

Ces diverses transactions démarrent donc par la saisie d'une adresse réticulaire (*étape 1 du schéma page suivante*) :



L'adresse saisie est un nom de domaine plus facile à retenir pour l'utilisateur et plus signifiant qu'une adresse IP (*Internet Protocol*). Cependant, pour communiquer entre les réseaux sur le Web on utilise l'adressage IP et non les noms de domaine.

Il va donc falloir aller rechercher (résoudre) quelle est l'adresse IP du site qui héberge `www.google.fr` (du domaine *google.fr*). On sollicite pour cela un serveur utilisant le protocole DNS (*Domain Name System*).



Le serveur DNS a parcouru ses enregistrements et a trouvé l'adresse IP associée au nom de domaine `www.google.fr` (*étape 2 du schéma ci-dessus*). S'il ne trouvait pas cette adresse dans ses enregistrements, il demanderait à un autre serveur ces renseignements (le serveur possède des « redirecteurs » vers d'autres serveurs DNS). De la même manière pour contrer l'éventuelle défaillance d'un serveur DNS on en déclare souvent deux sur une station.

Ayant maintenant l'adresse IP du site où risque de se trouver `www.google.fr`, la station va pouvoir poursuivre et solliciter les pages web (*étape 3 du schéma ci-dessus*). La station va faire une demande au serveur web via le protocole HTTP (en s'adressant à son port 80).

Pour pouvoir se synchroniser et être sûr que la page arrive bien dans l'ordre à la station : la station et le serveur vont échanger des informations et débiter une séquence de communication avec le protocole TCP (*Transmission Control Protocol*). Ce protocole est toujours associé au protocole IP quand on parle d'Internet. Ce protocole a donc le rôle d'encadrer les échanges Internet, comme ici notre navigation sur le Web, et assurer ainsi un transport des données plus fiable.

Le serveur web a répondu (*étape 4 du schéma ci-dessus*) à la demande en fournissant la page par défaut du site (`index.html` ou `index.php`). Suivant la taille du document HTML transporté, les données peuvent être

décomposées en plusieurs morceaux : des paquets. De façon simplifiée, on retiendra qu'un paquet est la plus petite unité circulant sur les réseaux étendus.

### A.1.3. Interactions web

Après avoir découvert le fonctionnement général du Web via le protocole HTTP, nous allons examiner comment interagissent les différentes composantes d'un serveur web.

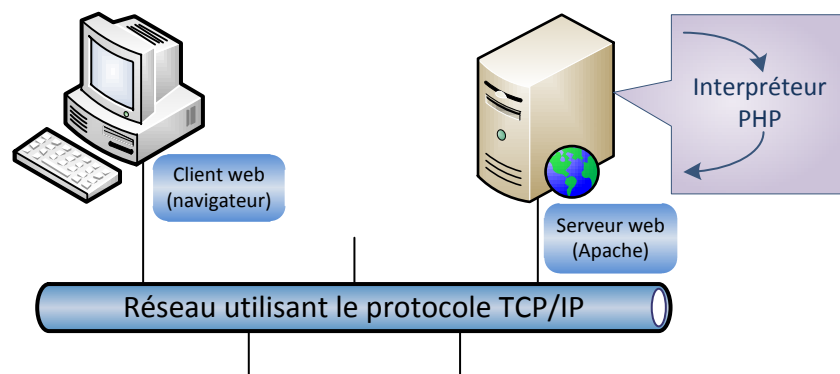
Vous avez jusqu'à présent utilisé un environnement de développement local très proche du fonctionnement réel des services web. La seule différence – et elle est de taille – est que tout se trouve situé sur votre poste localement. C'est à dire à la fois votre client web (le navigateur Firefox par exemple), votre service web (fourni par le serveur Apache), votre interpréteur PHP et votre base de données (MySQL, PostgreSQL) sont regroupés sur un même système.

Pour réaliser et tester une application web en PHP, vous devez disposer d'au moins 5 composants :

- Un réseau utilisant les protocoles TCP/IP et HTTP ;
- Un client web (typiquement un navigateur) qui demande l'interprétation d'un document ;
- Un serveur web (Apache, IIS, etc.) ;
- Un interpréteur PHP connu du serveur web ;
- Un document PHP, celui demandé par le client.

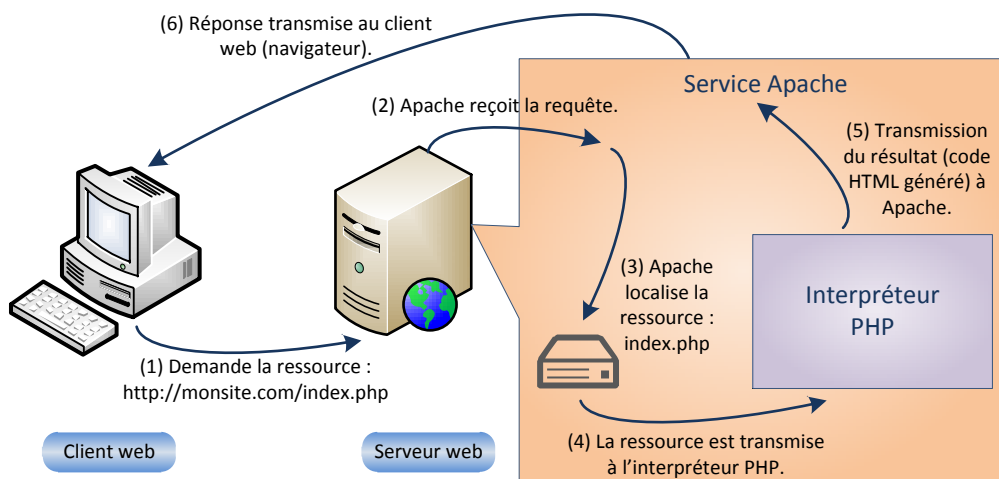
Dans le cadre d'une interaction avec une base de données il faudra également un moteur de base de données (MySQL ou PostgreSQL dans votre cas).

Lorsqu'il reçoit une requête HTTP référençant un document ayant l'extension **.php**, le serveur Apache le communique à un interpréteur PHP pour traitement.



Le résultat produit par l'interpréteur PHP est intercepté par Apache puis transmis au client à l'origine de la requête.

Apache se charge donc de localiser les ressources demandées (pages HTML, PHP, ressources diverses), lorsque celles-ci portent l'extension .php il les soumet à l'interpréteur PHP qui réalise les traitements demandés (ceux compris entre `<?php` et `?>`). L'interpréteur PHP transmet à son tour (si le développeur l'a prévu) du code HTML généré au service Apache.



## A.2. Eléments de base du langage PHP

Vous savez à présent que sans le service Apache qui fait appel à l'interpréteur PHP vous ne pourriez faire exécuter vos scripts PHP. Il est donc important que ces scripts soient stockés dans le répertoire racine (`www`) du serveur.

Nous allons découvrir la structure d'un document PHP orienté web. Il s'agit d'un document respectant en premier lieu les normes de présentation que nous avons adoptées pour nos documents XHTML. Par conséquent nous garderons bien en tête toutes ces connaissances.

### A.2.1. Structure et forme de base d'un script PHP

Le code PHP est encadré obligatoirement par `<?php ... ?>` qui délimite strictement ce code. C'est cette partie qui sera envoyée à l'interpréteur PHP côté serveur.

```
<?php echo '<?xml version="1.0" encoding="UTF-8" ?>' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Super document PHP</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  </head>
  <body>
    <?php echo '<h1>Voici le PHP !</h1>'; ?>
  </body>
</html>
```

# Voici le PHP !

Le code PHP contient l'instruction `echo` chargée de l'affichage (entre quotes), terminée par le classique « ; ». L'instruction génère du code HTML qui sera visible sur le client web.

L'usage des feuilles de style est donc possible de la même manière que pour n'importe quel document web.

```
<?php echo '<?xml version="1.0" encoding="UTF-8" ?>' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Super document</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <link rel="stylesheet" type="text/css" media="screen" href="css/fstyle.css" />
  </head>
  <body>
    <?php echo '<h1 class="bleu">Voici le PHP !</h1>'; ?>
  </body>
</html>
```

La feuille de style `fstyle.css` située dans le dossier `css` :

```
h1.bleu {
  color: #000099;
  font-family: verdana;
}
```

## Voici le PHP !

Malheureusement on perd tout l'intérêt de la colorisation syntaxique HTML avec le code inscrit dans l'instruction `echo`. Autant que possible le code HTML sera écrit tel quel (en dehors des commandes PHP) dans le document PHP.

```
<body>
  <h1 class="bleu">
    <?php echo 'Voici le PHP !'; ?>
  </h1>
</body>
```

On remarquera que comme pour le langage Javascript nous avons réservé les quotes au langage de programmation (autrement dit PHP) et les guillemets pour le langage HTML.

Cela aura pour incidence de devoir « échapper » les apostrophes lorsqu'elles sont contenues dans une instruction `echo` :

```
<body>
  <h1 class="bleu">
    <?php echo 'Voici le PHP et l\'échappement des apostrophes !'; ?>
  </h1>
</body>
```

## Voici le PHP et l'échappement des apostrophes !

Les lettres accentuées peuvent également être remplacées par leur code UNICODE.

```
<body>
  <h1 class="bleu">
    <?php echo 'Voici le PHP et l\'&eacute;chappement des apostrophes !'; ?>
  </h1>
</body>
```

### A.2.2. Structures de programmation

Vous connaissez déjà toutes les structures de programmation du langage PHP, il ne reste plus qu'à les utiliser judicieusement en vue de la production de contenus orientés web.

Par exemple pour construire un tableau HTML de 5 x 5 cases.

En HTML il faudrait une quarantaine de lignes et un certain temps de saisie (extrait du code) :

```
<body>
  <table>
    <tr>
      <td class="cellule">case 1</td>
      <td class="cellule">case 2</td>
      <td class="cellule">case 3</td>
      <td class="cellule">case 4</td>
      <td class="cellule">case 5</td>
    </tr>
    <tr>
      <td class="cellule">case 6</td>
      <td class="cellule">case 7</td>
      <td class="cellule">case 8</td>
      <td class="cellule">case 9</td>
      <td class="cellule">case 10</td>
    </tr>
    ...
  </table>
</body>
```

En PHP nous n'avons besoin que de 14 lignes :

```
<body>
  <table>
    <tr>
      <?php
        $i = 1;
        while ( $i < 26 ) {
          echo '<td class="cellule">case '.$i.'</td>';
          if ( ($i % 5 == 0) && ($i != 25) ) {
            echo '</tr><tr>';
          }
          $i++;
        }
      <?>
    </tr>
  </table>
</body>
```

Feuille de style utilisé :

```
td.cellule {  
    border: 1px solid #332211;  
    width: 50px;  
    height: 50px;  
}  
  
table {  
    border-collapse: collapse;  
}
```

Pour un résultat identique dans les deux cas :

case 1	case 2	case 3	case 4	case 5
case 6	case 7	case 8	case 9	case 10
case 11	case 12	case 13	case 14	case 15
case 16	case 17	case 18	case 19	case 20
case 21	case 22	case 23	case 24	case 25

Le code HTML qui a été généré (code source dans Firefox) :

```
<body>  
  <table>  
  
    <tr>  
      <td class="cellule">case 1</td><td class="cellule">case 2</td><td class="cellule">case  
3</td><td class="cellule">case 4</td><td class="cellule">case 5</td></tr><tr><td  
class="cellule">case 6</td><td class="cellule">case 7</td><td class="cellule">case 8</td><td  
class="cellule">case 9</td><td class="cellule">case 10</td></tr><tr><td class="cellule">case  
11</td><td class="cellule">case 12</td><td class="cellule">case 13</td><td class="cellule">case  
14</td><td class="cellule">case 15</td></tr><tr><td class="cellule">case 16</td><td  
class="cellule">case 17</td><td class="cellule">case 18</td><td class="cellule">case 19</td><td  
class="cellule">case 20</td></tr><tr><td class="cellule">case 21</td><td class="cellule">case  
22</td><td class="cellule">case 23</td><td class="cellule">case 24</td><td class="cellule">case  
25</td>  
      </tr>  
  
    </table>  
</body>
```

On peut faire trois remarques :

- l'instruction `echo` ne produit pas de saut de ligne ;
- on ne voit pas le code source PHP du document (le code encadré `<?php ... ?>`) ;
- seule l'instruction `echo` transmet les résultats des traitements PHP.



Remarque : Le code (doit être) commenté.

```
<body>
  <table> <!-- délimitation du tableau et de la première ligne directement en HTML -->
    <tr>
      <?php
        $i = 1; // initialisation de la variable servant à la construction du tableau
        while ( $i < 26 ) { // boucle allant de 1 à 25
          echo '<td class="cellule">case '.$i.'</td>'; // affichage d'une cellule de
tableau avec son numéro
          if ( ($i % 5 == 0) && ($i != 25) ) { // changement de ligne toutes les 5 lignes,
sauf à la dernière
            echo '</tr><tr>'; // fin de ligne et début d'une nouvelle
          }
          $i++; // incrémentation de la variable
        }
      <?>
    </tr> <!-- délimitation de la dernière ligne, puis fermeture du tableau -->
  </table>
</body>
```

Grâce au langage de programmation, on peut donc impacter la présentation même du tableau, par exemple en alternant case jaunes et cases bleues, avec une condition :

```
<body>
  <table>
    <tr>
      <?php
        $i = 1;
        while ( $i < 26 ) {
          if ( $i % 2 == 0 ) { // lorsque le numéro de cellule est pair
            echo '<td class="cellbleue">case '.$i.'</td>';
          }
          else { // cas contraire : lorsque le numéro de cellule est impair
            echo '<td class="celljaune">case '.$i.'</td>';
          }

          if ( ($i % 5 == 0) && ($i != 25) ) {
            echo '</tr><tr>';
          }
          $i++;
        }
      <?>
    </tr>
  </table>
</body>
```

```
td.cellbleue, td.celljaune {
  border: 1px solid #332211;
  width: 50px;
  height: 50px;
  text-align: center;
}

td.cellbleue {
  background-color: #99FFFF;
}

td.celljaune {
  background-color: #FFFF66;
}

table {
  border-collapse: collapse;
}
```

case 1	case 2	case 3	case 4	case 5
case 6	case 7	case 8	case 9	case 10
case 11	case 12	case 13	case 14	case 15
case 16	case 17	case 18	case 19	case 20
case 21	case 22	case 23	case 24	case 25

### A.2.3. Mise en page avec PHP

Le langage PHP est imbattable pour la structuration des pages web.

Nous avons vu, à l'occasion de l'utilisation des feuilles de style, qu'un document HTML type comprenait diverses parties disposées grâce à des balises de bloc (<div></div> ou <p></p> par exemple).

En HTML le codage est « statique » :

```
<body>
  <div class="entete">
    Le site : tout en PHP... ou presque
  </div>
  <div class="menu">
    <b>menu :</b><br/>
    - Aller chez <a href="http://www.google.fr">google</a><br/>
    - Aller à <a href="mapage1.html">mapage 1</a><br/>
    - et puis voilà...
  </div>
  <div class="centre">
    Ici il y a du texte, et des choses intéressantes, des bidules.<br/> Et du blabla.
  </div>
</body>
```

```
a:link, a:visited {
  text-decoration: none;
  color: orange;
}

a:hover {
  background: black;
  color: yellow;
  text-decoration: overline underline;
}

div.entete {
  font-family: verdana;
  font-size: 24pt;
  position: absolute;
  top: 10px;
  left: 10px;
  width: 700px;
  height: 70px;
  padding-top: 20px;
  background-color: #CCCC99;
  text-align: center;
  border-radius: 5px;
}
```

```
div.menu {
  position: absolute;
  top: 110px;
  left: 10px;
  width: 150px;
  height: 400px;
  padding: 5px;
  background-color: #CCFFCC;
  border-radius: 5px;
}

div.centre {
  position: absolute;
  top: 110px;
  left: 180px;
  width: 520px;
  height: 400px;
  padding: 5px;
  background-color: #FF9933;
  border-radius: 5px;
}
```

# Le site : tout en PHP... ou presque

## menu :

- Aller chez [google](#)
- Aller à [mapage 1](#)
- et puis voilà...

Ici il y a du texte, et des choses intéressantes, des bidules.  
Et du blabla.

Le PHP va offrir la possibilité d'inclure des portions de code HTML ou PHP dans une page centralisatrice (idéalement la page *index.php*). Cela permet une meilleure lisibilité du code, mais surtout de pas avoir à gérer les changements de code dans toutes les pages ayant la même présentation. Autrement dit chaque portion de code peut être incluse dans différentes pages et les évolutions du code faites sur ces portions le seront pour toutes les pages.

En PHP avec la fonction `include()` on peut « inclure » des portions de code HTML ou PHP :

```
<body>
  <?php include 'includes/entete.inc.php'?>
  <?php include 'includes/menu.inc.php'?>
  <div class="centre">
    Ici il y a du texte, et des choses intéressantes, des bidules.<br/> Et du blabla.
  </div>
</body>
```

Le fichier *entete.inc.php* :

```
<div class="entete">
  Le site : tout en PHP... ou presque
</div>
```

Le fichier *menu.inc.php* :

```
<div class="menu">
  <b>menu :</b><br/>
  - Aller chez <a
href="http://www.google.fr">google</a><br/>
  - Aller à <a href="mapage1.html">mapage
1</a><br/>
  - et puis voilà...
</div>
```

Le résultat sera rigoureusement identique, avec la possibilité dorénavant de pouvoir changer les diverses portions de code de façon indépendante. Le nommage en *.inc.php* des fichiers inclus est une convention de nommage qu'il vaut mieux adopter.

### A.2.1. Utilisation de fonctions et de procédures

Les fonctions et procédures PHP sont très utiles pour pouvoir répéter certains traitements. Elles peuvent être incluses à partir d'un fichier externe et être appelées selon les besoins. On notera que fonctions et procédures sont toutes déclarées par l'instruction **function**.

Par exemple, nous pourrions mettre sous forme de fonction notre création de tableau de 5 x 5 cases et l'écrire dans un fichier externe.

Le fichier PHP *fonctions.inc.php* contenant la fonction **tableau()** :

```
<?php
/**
 * Fonction permettant de construire un tableau de 5 x 5 cases.
 * Les cases sont alternées bleues (cases paires) et jaunes (cases impaires)
 */

function tableau() {
    echo '<table>';
    echo '<tr>';
    $i = 1;
    while ( $i < 26 ) {
        if ( $i % 2 == 0 ) {
            echo '<td class="cellbleue">case '.$i.'</td>';
        }
        else {
            echo '<td class="celljaune">case '.$i.'</td>';
        }

        if ( ($i % 5 == 0) && ($i != 25) ) {
            echo '</tr><tr>';
        }
        $i++;
    }
    echo '</tr>';
    echo '</table>';
}
?>
```

#### Remarques :

- La fonction **tableau()** ne contient que des instructions PHP.
- En début de fichier on trouve une documentation sur la fonction. Il s'agit d'un bloc documentaire, qui est un bloc de commentaires étendu qui commence par un **/\*\*** et présente un **\*** au début de chaque ligne. Les blocs de commentaires précèdent les éléments qu'ils documentent. C'est une bonne habitude à prendre de façon à ce que le code puisse être repris par une tierce personne ou être remanié plus tard.

La page qui inclus le fichier et utilise la fonction :

```
<body>
<?php include_once 'includes/fonctions.inc.php'; ?>
<?php include 'includes/entete.inc.php'; ?>
<?php include 'includes/menu.inc.php'; ?>
<div class="centre">
    Ici il y a du texte, et des choses intéressantes, des bidules.<br/>
    Et un beau tableau :<br/>
    <?php tableau(); ?>
</div>
</body>
```

Pour inclure le fichier PHP *fonctions.inc.php* nous avons utilisé l'instruction `include_once` qui assure que le code du fichier inclus ne sera ajouté qu'une seule fois, évitant de ce fait les redéfinitions de variables ou de fonctions, génératrices d'alertes. En effet si par hasard vous « incluez » plusieurs fois une même fonction ou des mêmes variables celles-ci ne le seront qu'une fois.

## Le site : tout en PHP... ou presque

### menu :

- Aller chez [google](#)
- Aller à [mapage 1](#)
- et puis voilà...

Ici il y a du texte, et des choses intéressantes, des bidules.  
Et un beau tableau :

case 1	case 2	case 3	case 4	case 5
case 6	case 7	case 8	case 9	case 10
case 11	case 12	case 13	case 14	case 15
case 16	case 17	case 18	case 19	case 20
case 21	case 22	case 23	case 24	case 25

Les fonctions et d'une façon plus générale la programmation PHP permettent de rendre les contenus dynamiques et de générer du contenu HTML adapté aux traitements. Pour l'instant nos scripts se contentent de structurer l'affichage.

Notre tableau comporte uniquement 5 x 5 cases. Avec un paramètre, en adaptant la fonction on peut lui faire prendre n'importe quelle taille.

Le fichier PHP *fonctions.inc.php* contenant la fonction **tableau(\$nb)** :

```
<?php
/**
 * Fonction permettant de construire un tableau de $nb x $nb cases.
 * Les cases sont alternées bleues (cases paires) et jaunes (cases impaires)
 */

function tableau($nb) {
    if ( $nb > 10 ) { // limitation de la taille du tableau à dix cases uniquement
        $nb = 10;
    }
    echo '<table>';
    echo '<tr>';
    $i = 1;
    while ( $i <= $nb*$nb ) { // le tableau fait $nb x $nb cases
        if ( $i % 2 == 0 ) {
            echo '<td class="cellbleue">case '.$i.'</td>';
        }
        else {
            echo '<td class="celljaune">case '.$i.'</td>';
        }

        if ( ($i % $nb == 0) && ($i != $nb*$nb) ) {
            // on change de ligne toute les $nb cases sauf à la fin
            echo '</tr><tr>';
        }
        $i++;
    }
    echo '</tr>';
    echo '</table>';
}
?>
```

```
<body>
    <?php include_once 'includes/fonctions.inc.php'; ?>
    <?php include 'includes/entete.inc.php'; ?>
    <?php include 'includes/menu.inc.php'; ?>
    <div class="centre">
        Ici il y a du texte, et des choses intéressantes, des bidules.<br/> Et un beau tableau
    :<br/>
    <?php tableau('7'); ?>
    </div>
</body>
```

Remarque : Ici la valeur envoyée comme paramètre est « 7 ».

**menu :**

- Aller chez [google](#)
- Aller à [mapage 1](#)
- et puis voilà...

Ici il y a du texte, et des choses intéressantes, des bidules.  
Et un beau tableau :

case 1	case 2	case 3	case 4	case 5	case 6	case 7
case 8	case 9	case 10	case 11	case 12	case 13	case 14
case 15	case 16	case 17	case 18	case 19	case 20	case 21
case 22	case 23	case 24	case 25	case 26	case 27	case 28
case 29	case 30	case 31	case 32	case 33	case 34	case 35
case 36	case 37	case 38	case 39	case 40	case 41	case 42
case 43	case 44	case 45	case 46	case 47	case 48	case 49

Il sera donc ensuite possible d'entrer d'autres dimensions aux tableaux où hauteur et largeur ne seront plus identiques (tableau rectangulaire).

```
<?php
/**
 * Fonction permettant de construire un tableau de $largeur x $hauteur cases.
 * Les cases sont alternées bleues (cases paires) et jaunes (cases impaires)
 */

function tableau($largeur, $hauteur) {
    if ( $largeur > 10 ) {
        $largeur = 10;
    }
    if ( $hauteur > 10 ) {
        $hauteur = 10;
    }
    echo '<table>';
    echo '<tr>';
    $i = 1;
    while ( $i <= $largeur*$hauteur ) {
        if ( $i % 2 == 0 ) {
            echo '<td class="cellbleue">case '.$i.'</td>';
        }
        else {
            echo '<td class="celljaune">case '.$i.'</td>';
        }

        if ( ($i % $largeur == 0) && ($i != $largeur*$hauteur) ) {
            echo '</tr><tr>';
        }
        $i++;
    }
    echo '</tr>';
    echo '</table>';
}
?>
```

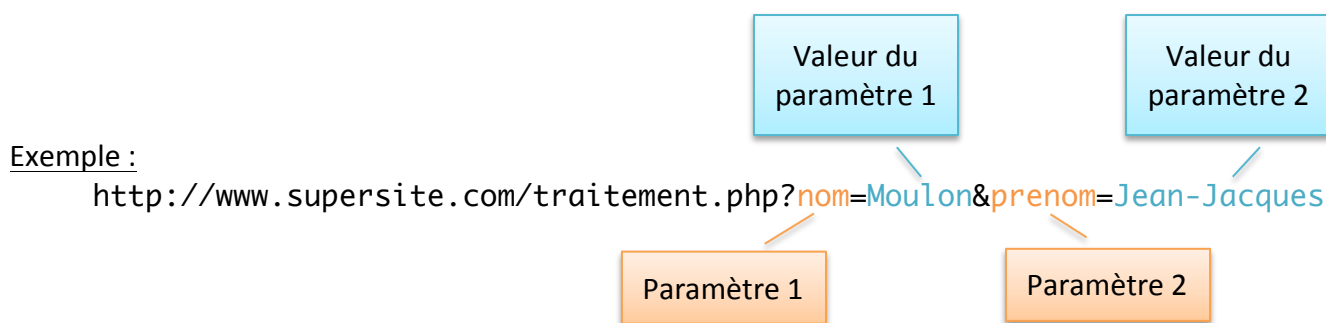
## B. La transmission des données

### B.1. Transmission des données passées par l'URL

#### B.1.1. Principe

L'exécution côté serveur d'un programme PHP est déclenchée lors d'une requête lancée par le client web. Par exemple la requête `http://www.supersite.com/traitement.php` déclenche une action, appelée GET, côté serveur. C'est typiquement une demande de consultation de ressources (get => obtenir). Dans cet exemple, aucune donnée n'est transmise.

Dans le cas où des données sont transmises en même temps que cette requête, ces données sont alors disponibles **depuis** l'application PHP via une collection nommée `$_GET` ; Les couples (clé, valeur) passés sur l'URL constituent les éléments de `$_GET`.

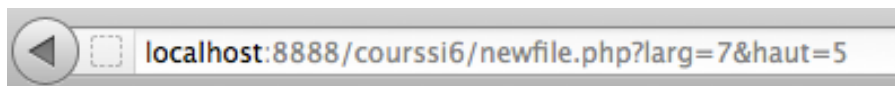


Le programme PHP pourra récupérer ces données via la collection `$_GET`, en lui passant comme valeur de clé le nom de la clé de l'argument, soit dans notre cas : `$_GET['nom']` et `$_GET['prenom']`.

L'inconvénient majeur de cette méthode est son affichage dans l'URL et donc la possibilité de modifier les données transmises.

#### B.1.2. Passage de paramètre simple

Exemple : données saisies dans l'URL permettant de dimensionner du tableau (à écrire dans l'URL).



```
<body>
<?php include_once 'includes/fonctions.inc.php'; ?>
<?php include 'includes/entete.inc.php'; ?>
<?php include 'includes/menu.inc.php'; ?>
<div class="centre">
    Ici il y a du texte, et des choses intéressantes, des bidules.<br/> Et un beau tableau
:<br/>
    <?php tableau($_GET['larg'],$_GET['haut']); ?>
</div>
</body>
```



```

<?php
function tableau($largeur, $hauteur) {
    if ( $largeur > 10 ) {
        $largeur = 10;
    }
    if ( $hauteur > 10 ) {
        $hauteur = 10;
    }
    echo '<table>';
    echo '<tr>';
    $i = 1;
    while ( $i <= $largeur*$hauteur ) {
        if ( $i % 2 == 0 ) {
            echo '<td class="cellbleue">case '.$i.'</td>';
        }
        else {
            echo '<td class="celljaune">case '.$i.'</td>';
        }

        if ( ($i % $largeur == 0) && ($i != $largeur*$hauteur) ) {
            echo '</tr><tr>';
        }
        $i++;
    }
    echo '</tr>';
    echo '</table>';
}
?>

```

localhost:8888/courssi6/newfile.php?larg=7&haut=5

## Le site : tout en PHP... ou presque

### menu :

- Aller chez [google](#)
- Aller à [mapage 1](#)
- et puis voilà...

Ici il y a du texte, et des choses intéressantes, des bidules.

Et un beau tableau :

case 1	case 2	case 3	case 4	case 5	case 6	case 7
case 8	case 9	case 10	case 11	case 12	case 13	case 14
case 15	case 16	case 17	case 18	case 19	case 20	case 21
case 22	case 23	case 24	case 25	case 26	case 27	case 28
case 29	case 30	case 31	case 32	case 33	case 34	case 35

### B.1.3. Le contrôle des paramètres

L'inconvénient de passer un paramètre par l'URL est notamment qu'il peut être modifié ou effacé par le visiteur. Si un paramètre est absent cela peut provoquer une erreur révélant alors le code PHP en cause.

Exemple, une donnée est absente de l'URL non prévue dans le code PHP.



Il manque le paramètre haut=... et par conséquent le tableau ne s'affiche pas. Et parfois cela peut déclencher l'affichage d'une erreur de la part de PHP, ce qui peut être gênant, voir dramatique...

La fonction `isset()` sert à vérifier qu'une variable existe et permet de déclencher (ou non) un traitement particulier sans générer d'erreur propre à l'interpréteur PHP (parfois grave). Vous pouvez donc gérer vous même les erreurs, les prévenir et afficher au visiteur des messages adaptés.

Exemple, une donnée est absente de l'URL et détectée dans code PHP.

```
<body>
<?php include_once 'includes/fonctions.inc.php'; ?>
<?php include 'includes/entete.inc.php'; ?>
<?php include 'includes/menu.inc.php'; ?>
<div class="centre">
    Ici il y a du texte, et des choses intéressantes, des bidules.<br/>
    <?php
        if ( isset($_GET['larg']) && isset($_GET['haut']) ) {
            echo 'Et un beau tableau :<br/>';
            tableau($_GET['larg'],$_GET['haut']);
        }
        else {
            echo '<h4 class="error">Erreur, paramètre absent</h4>';
        }
    ?>
</div>
</body>
```

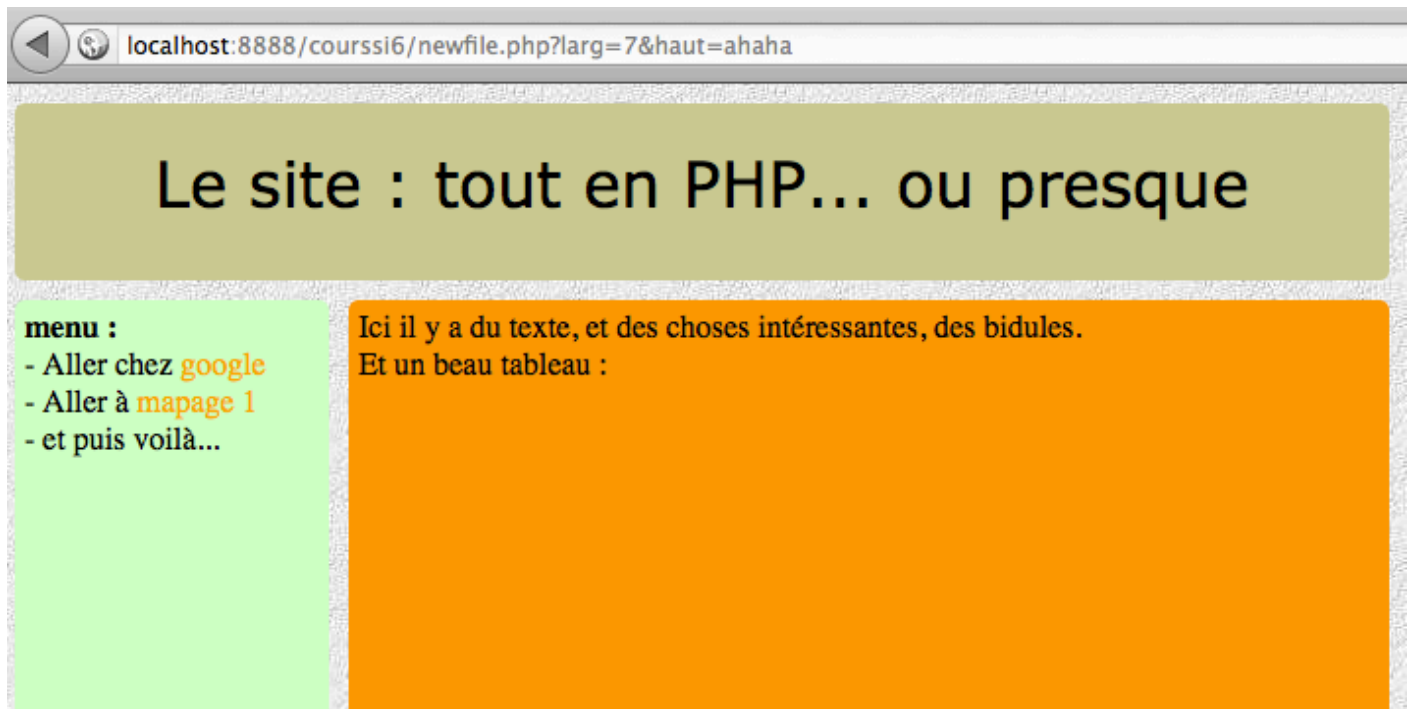


En plus du contrôle de la présence de toutes les données saisies dans l'URL il faut s'assurer qu'elles soient adaptées (au bon format) et dans la zone de définition prévue (valeurs aberrantes exclues).

Le **transtypage** permet de transformer des données saisies dans un format dans un autre. Par exemple, on attend un nombre entier et ce sont des caractères qui sont mis. Il faudra alors transformer les données saisies en nombre entier (ce qui donnera la valeur 0 aux caractères saisis). L'objectif est d'éviter le déclenchement d'une erreur de la part de l'interpréteur PHP.

Exemple, contrôle du contenu des données saisies dans l'URL

```
<body>
<?php include_once 'includes/fonctions.inc.php'; ?>
<?php include 'includes/entete.inc.php'; ?>
<?php include 'includes/menu.inc.php'; ?>
<div class="centre">
    Ici il y a du texte, et des choses intéressantes, des bidules.<br/>
    <?php
        if ( isset($_GET['larg']) && isset($_GET['haut']) ) {
            // transtypage de $_GET['larg'] et $_GET['haut']
            // en entier pour éviter la saisie de caractères
            $_GET['larg'] = (int) $_GET['larg'];
            $_GET['haut'] = (int) $_GET['haut'];
            echo 'Et un beau tableau :<br/>';
            tableau($_GET['larg'],$_GET['haut']);
        }
        else {
            echo '<h4 class="error">Erreur, paramètre absent</h4>';
        }
    }
    ?>
</div>
</body>
```



Le paramètre `haut=ahaha` a été transformé en entier et a pris par conséquent la valeur 0. Le tableau comportant la valeur 0 comme hauteur ne s'est pas dessiné.

## B.2. Transmission des données passées par formulaire

Les types de requêtes les plus fréquemment utilisés par le client sont de type GET et POST. La méthode GET est conçue pour obtenir des informations (document, résultat d'une requête sur une base de données) auprès d'une application Web.

On a généralement recours au **formulaire HTML** pour envoyer des données au serveur.

### B.2.1. Transmission des données avec la méthode GET

Exemple : un formulaire HTML permet de saisir son nom et son prénom et de l'envoyer à une page d'inscription (nommée *inscription.php*).

```
<body>
  <p>Votre inscription</p>
  <form id="f1" method="get" action="inscrire.php">
    <fieldset>
      <legend>Identité :</legend>
      Entrez votre nom <input class="question" type="text" id="nom" name="nom" /><br/><br/>
      Entrez votre prénom <input class="question" type="text" id="prenom" name="prenom" />
      <br/><br/>
      <input name="b1" type="reset" value="effacer" /><input name="b2" type="submit"
        value="valider" />
    </fieldset>
  </form>
</body>
```

## Votre inscription

Identité :

Entrer votre nom

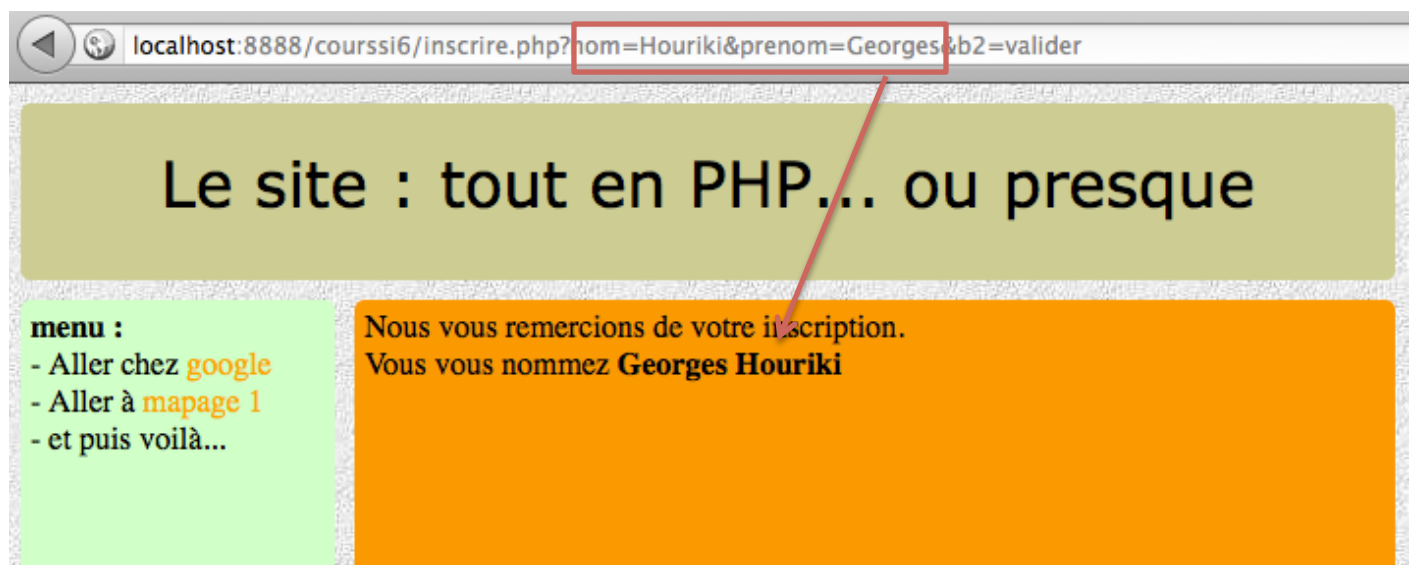
Entrer votre prénom

Le script PHP *inscription.php* reçoit les paramètres via l'URL.

```
<body>
<?php include 'includes/entete.inc.php'; ?>
<?php include 'includes/menu.inc.php'; ?>
<div class="centre">
  <?php
    if ( isset($_GET['nom']) && isset($_GET['prenom']) ) {
      echo 'Nous vous remercions de votre inscription.<br/>';
      echo 'Vous vous nommez <b>'. $_GET['prenom']. ' ' . $_GET['nom']. '</b>';
    }
    else {
      echo '<h4 class="error">Erreur, vous n\'avez pas saisi tous les champs !</h4>';
    }
  ?>
</div>
</body>
```

On voit apparaître dans l'URL `nom=Houriki&prenom=Georges&b2=valider` qui correspond aux éléments du formulaire :

- `<input class="question" type="text" id="nom" name="nom" />`
- `<input class="question" type="text" id="prenom" name="prenom" />`
- `<input name="b2" type="submit" value="valider" />`



Remarque : Il est intéressant de noter que même le bouton qui a été utilisé pour valider le formulaire est transmis au script *inscrire.php*. Cela permet, lorsque plusieurs formulaires ou différentes actions sont proposées (par exemple « éditer », « ajouter », « supprimer », etc.), de les distinguer et d'effectuer des traitements adaptés.

## B.2.2. Transmission des données avec la méthode POST

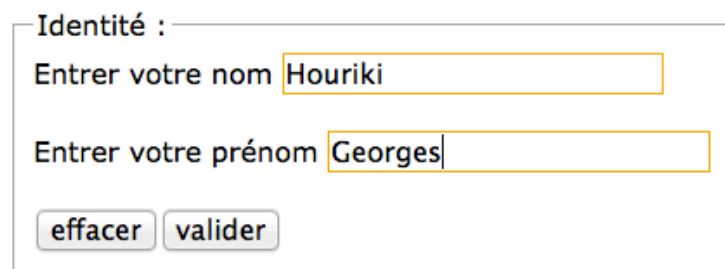
La méthode POST utilise une technique différente pour envoyer des données au serveur. Les données sont placées directement dans le corps de la requête et n'apparaissent donc pas dans la chaîne d'interrogation de l'URL.

Cette méthode permet d'envoyer autant de données que l'on veut, ce qui fait qu'on la privilégie le plus souvent. Néanmoins, les données ne sont pas plus sécurisées qu'avec la méthode GET et il faudra toujours vérifier si tous les paramètres sont bien présents et valides comme avec la méthode GET. **On doit se méfier des données provenant de formulaires autant que celles provenant de l'URL.**

### B.2.2.1. Utilisation de la méthode POST

```
<body>
  <p>Votre inscription</p>
  <form id="f1" method="post" action="inscrire.php">
    <fieldset>
      <legend>Identité :</legend>
      Entrer votre nom <input class="question" type="text" id="nom" name="nom" /><br/><br/>
      Entrer votre prénom <input class="question" type="text" id="prenom" name="prenom" />
      <br/><br/>
      <input name="b1" type="reset" value="effacer" /><input name="b2" type="submit"
        value="valider" />
    </fieldset>
  </form>
</body>
```

Votre inscription



Le script PHP *inscription.php* reçoit les paramètres **directement dans le corps de la requête.**

```
<body>
  <?php include 'includes/entete.inc.php'; ?>
  <?php include 'includes/menu.inc.php'; ?>
  <div class="centre">
    <?php
      if ( isset($_POST['nom']) && isset($_POST['prenom']) ) {
        $lenom = $_POST['nom'];
        $leprenom = $_POST['prenom'];
        echo 'Nous vous remercions de votre inscription.<br/>';
        echo 'Vous vous nommez <b>'.$leprenom.' '.$lenom.'</b>';
      }
      else {
        echo '<h4 class="error">Erreur, vous n\'avez pas saisi tous les champs !</h4>';
      }
    <?>
  </div>
</body>
```



#### Remarque :

- Les données sont transmises sans apparaître dans l'URL.
- cette fois-ci les données réceptionnées ont été placées dans des variables. Cela permet de pouvoir centraliser et référencer toutes les données reçues et de pouvoir permettre le passage d'une méthode à une autre.
  - \$lenom = \$\_POST['nom'];
  - \$leprenom = \$\_POST['prenom'];

#### B.2.2.2. Utilisation d'une case à cocher dans un formulaire

```
<body>
  <p>Votre inscription</p>
  <form id="f1" method="post" action="inscrire.php">
    <fieldset>
      <legend>Identité :</legend>
      Entrer votre nom <input class="question" type="text" id="nom" name="nom" /><br /><br />
      Entrer votre prénom <input class="question" type="text" id="prenom" name="prenom" />
      <br /><br />
      Vous certifiez avoir lu notre charte <input type="checkbox" id="certif" name="certif" />
      <br /><br />
      <input name="b1" type="reset" value="effacer" /><input name="b2" type="submit"
        value="valider" />
    </fieldset>
  </form>
</body>
```

#### Votre inscription

Identité :
 

Entrer votre nom

Entrer votre prénom

Vous certifiez avoir lu notre charte
 ☐

Le script PHP *inscription.php* reçoit les paramètres.

```
<body>
<?php include 'includes/entete.inc.php'; ?>
<?php include 'includes/menu.inc.php'; ?>
<div class="centre">
  <?php
    if ( isset($_POST['nom']) && isset($_POST['prenom']) ) {
      $lenom = $_POST['nom'];
      $leprenom = $_POST['prenom'];
      if ( isset($_POST['certif']) ) {
        echo 'Nous vous remercions de votre inscription.<br/>';
        echo 'Vous vous nommez <b>'.$leprenom.' '.$lenom.'</b>';
      }
      else {
        echo 'Désolé '.$leprenom.' '.$lenom.', vous devez certifier avoir lu notre
          charte !';
      }
    }
    else {
      echo '<h4 class="error">Erreur, vous n\'avez pas saisi tous les champs !</h4>';
    }
  ?>
</div>
</body>
```

## Le site : tout en PHP... ou presque

### menu :

- Aller chez [google](#)
- Aller à [mapage 1](#)
- et puis voilà...

Désolé Georges Houriki, vous devez certifier avoir lu notre charte !

Remarque : Une case à cocher non utilisée n'existe pas ! En testant la présence de `$_POST['certif']` on sait si la case a été cochée ou non.



### B.2.2.3. Utilisation de boutons radio dans un formulaire

```
<body>
<p>Votre inscription</p>
<form id="f1" method="post" action="inscrire.php">
<fieldset>
<legend>Identité :</legend>
Mme <input type="radio" id="titre" name="titre" value="Madame" checked="checked" />
Mlle <input type="radio" id="titre" name="titre" value="Mademoiselle" />
M. <input type="radio" id="titre" name="titre" value="Monsieur" /><br /><br />
Entrer votre nom <input class="question" type="text" id="nom" name="nom" /><br /><br />
Entrer votre prénom <input class="question" type="text" id="prenom" name="prenom" />
<br /><br />
Vous certifiez avoir lu notre charte <input type="checkbox" id="certif" name="certif" />
<br /><br />
<input name="b1" type="reset" value="effacer" /><input name="b2" type="submit"
value="valider" />
</fieldset>
</form>
</body>
```

#### Votre inscription

Identité : —

Mme ☐ Mlle ☐ M. ☒

Entrer votre nom

Entrer votre prénom

Vous certifiez avoir lu notre charte ☒

Le script PHP *inscription.php* reçoit les paramètres.

```
<body>
<?php include 'includes/entete.inc.php'; ?>
<?php include 'includes/menu.inc.php'; ?>
<div class="centre">
<?php
if ( isset($_POST['nom']) && isset($_POST['prenom']) && isset($_POST['titre']) ) {
    $letitre = $_POST['titre'];
    $lenom = $_POST['nom'];
    $leprenom = $_POST['prenom'];
    if ( isset($_POST['certif']) ) {
        echo 'Nous vous remercions de votre inscription.<br/>';
        echo 'Vous êtes <b>'.$letitre.' '.$leprenom.' '.$lenom.'</b>';
    }
    else {
        echo 'Désolé '.$letitre.' '.$leprenom.' '.$lenom.', vous devez certifier avoir
        lu notre charte !';
    }
}
else {
    echo '<h4 class="error">Erreur, vous n\'avez pas saisi tous les champs !</h4>';
}
?>
</div>
</body>
```

# Le site : tout en PHP... ou presque

## menu :

- Aller chez **google**
- Aller à **mapage 1**
- et puis voilà...

Nous vous remercions de votre inscription.

Vous êtes **Monsieur Georges Houriki**

## Remarques :

- Les boutons radio portent un nom unique par groupe, et ont chacun une valeur qui leur est propre. Ici la valeur de chacun des boutons n'est pas identique à ce qui est proposé au visiteur. Seule la valeur inscrite dans **value** est transmise.
- L'usage est identique pour les listes déroulantes.

## B.2.3. Page qui s'appelle elle-même

Un formulaire peut adresser la page qui le contient elle-même. C'est utile quand on veut faire des traitements en cascade, par exemple où un premier formulaire permet d'en compléter un autre. C'est également utile lorsque l'on veut centraliser dans un seul script des traitements identiques sur un même type de données.

Exemple : un formulaire HTML permet de saisir son nom et son prénom et de l'envoyer à la page elle-même :

Le script contiendra deux parties :

```
if ( isset($_POST['b2']) ) {  
    // Le bouton b2 a été pressé, les données sont traitées et affichées  
}  
else {  
    // Le bouton n'a pas été pressé, le formulaire est affiché  
}
```

## Remarques :

- Le contrôle **isset()** sert à filtrer le cas où l'on a déjà utilisé le formulaire (en ce cas \$\_POST['b2'] est affecté).
- Le formulaire n'est affiché que dans le cas où l'on n'a pas pressé sur le bouton b2.

```

<body>
<?php include 'includes/entete.inc.php'; ?>
<?php include 'includes/menu.inc.php'; ?>
<div class="centre">
  <?php
    if ( isset($_POST['b2']) ) {
      if ( isset($_POST['nom']) && isset($_POST['prenom']) && isset($_POST['titre']) ) {
        $letitre = $_POST['titre'];
        $lenom = htmlspecialchars($_POST['nom'], ENT_NOQUOTES);
        $leprenom = htmlspecialchars($_POST['prenom'], ENT_NOQUOTES);
        echo 'Nous vous remercions de votre inscription.<br/>';
        echo 'Vous êtes <b>'.$letitre.' '.$leprenom.' '.$lenom.'</b>';
      }
      else {
        echo '<h4 class="error">Erreur, vous n\'avez pas saisi tous les champs !</h4>';
      }
    }
    else {
  ?>
  <p>Votre inscription</p>
  <form id="f1" method="post" action="inscription.php">
  <fieldset>
    <legend>Identité :</legend>
    Mme <input type="radio" id="titre" name="titre" value="Madame" checked="checked"
    />
    Mlle <input type="radio" id="titre" name="titre" value="Mademoiselle" />
    M. <input type="radio" id="titre" name="titre" value="Monsieur" /><br /><br />
    Entrer votre nom <input class="question" type="text" id="nom" name="nom"
    /><br/><br/>
    Entrer votre prénom <input class="question" type="text" id="prenom" name="prenom"
    /><br/><br/>
    <input name="b1" type="reset" value="effacer" /><input name="b2" type="submit"
    value="valider" />
  </fieldset>
  </form>

  <?php
  }
  ?>
</div>
</body>

```

localhost:8888/courssi6/inscription.php

## Le site : tout en PHP... ou presque

**menu :**

- Aller chez google
- Aller à mapage 1
- et puis voilà...

**Votre inscription**

Identité :

Mme ☐ Mlle ☐ M. ☒

Entrer votre nom

Entrer votre prénom



Remarque : le code du formulaire a été mis hors du code PHP puisqu'il n'était pas utile ici. On remarque qu'après l'accolade du `else` le code est arrêté :

```
else {  
?>
```

Il reprend après le formulaire pour fermer l'accolade ouverte avec `else` :

```
<?php  
}  
?>
```

#### B.2.4. Sécurisation des données transmises

Toute donnée saisie à l'aide d'un formulaire peut engendrer des dysfonctionnements, que ce soit lors de l'enregistrement dans la base de données ou lors de son affichage ultérieur dans une page HTML. En effet certains caractères, notamment `<` et `>` qui sont les délimiteurs des balises, peuvent provoquer des erreurs parfois critiques ou pire permettre l'injection de code HTML contenant du Javascript.

La faille XSS (pour cross-site scripting) permet d'injecter du code HTML contenant du Javascript dans vos pages pour le faire exécuter aux visiteurs.

Imaginez qu'à la place de saisir son nom un visiteur saisisse le code suivant :

Entrer votre nom

Rien de « grave », son nom s'affichera désormais en gros lorsqu'il sera réutilisé dans du code et interprété en HTML :

Nous vous remercions de votre inscription.  
Vous êtes **Monsieur Georges**

**Houriki**

Maintenant imaginons que cet utilisateur est mal intentionné et saisisse le code suivant :

Entrer votre nom

Ou encore (il y a pire encore... ici les exemples sont volontairement « gentils ») :

Entrer votre nom

Pour éviter ces effets détournements, les caractères réservés doivent être traduits en symboles nommés HTML (aussi appelés entités HTML). Ainsi, le caractère < doit être transformé en &lt;;, > doit être transformé en &gt;;, etc.

La fonction PHP prédéfinie `htmlspecialchars()` prend en charge ce traitement.

Les remplacements effectués sont :

- " & " (et commercial) devient " &amp; " ;
- " " " (guillemets doubles) devient " &quot; " ;
- " ' " (single quote) devient " &#039; " ;
- " < " (supérieur à) devient " &lt; " ;
- " > " (inférieur à) devient " &gt; " .

Dans ce cas la saisie suivante :

Votre inscription

Identité :

Mme ☐ Mlle ☐ M. ☒

Entrer votre nom

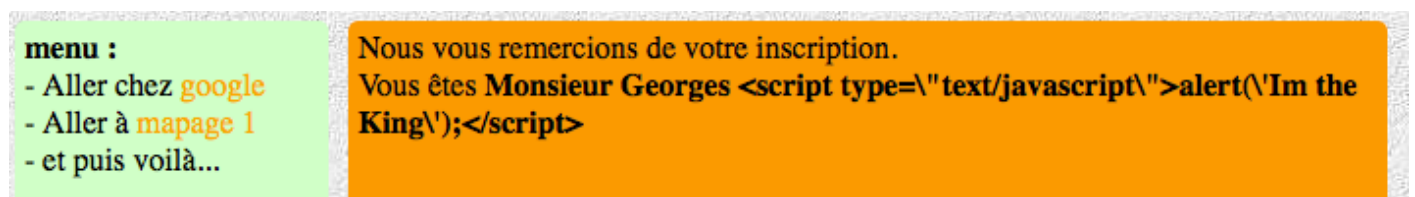
Entrer votre prénom

Vous certifiez avoir lu notre charte ☒

Avec le script *inscription.php* utilisant la fonction `htmlspecialchars()` :

```
<body>
<?php include 'includes/entete.inc.php'; ?>
<?php include 'includes/menu.inc.php'; ?>
<div class="centre">
  <?php
    if ( isset($_POST['nom']) && isset($_POST['prenom']) && isset($_POST['titre']) ) {
      $letitre = $_POST['titre'];
      $lenom = htmlspecialchars($_POST['nom']);
      $leprenom = htmlspecialchars($_POST['prenom']);
      if ( isset($_POST['certif']) ) {
        echo 'Nous vous remercions de votre inscription.<br/>';
        echo 'Vous êtes <b>'.$letitre.' '.$leprenom.' '.$lenom.'</b>';
      }
      else {
        echo 'Désolé '.$letitre.' '.$leprenom.' '.$lenom.', vous devez certifier avoir lu
          notre charte !';
      }
    }
    else {
      echo '<h4 class="error">Erreur, vous n\'avez pas saisi tous les champs !</h4>';
    }
  }
  ?>
</div>
</body>
```

Rendra visible code saisi à la place du nom, puisque les caractères < et > spécifiques aux balises HTML seront traduits en symboles nommés.



Remarque : Une grande partie des contrôles sur formulaire peuvent être réalisés en Javascript et donc exécutés sur le client web directement. Cela permet de dégager le serveur de la charge des contrôles pour lui donner (exclusivement) les traitements à réaliser.

## B.3. La transmission des fichiers

Les formulaires ont également pour rôle la transmission des fichiers. Le rôle des sites n'est pas uniquement de gérer des informations descendantes mais également des données ascendantes. Nous le verrons, les bases de données s'alimentent des données provenant des visiteurs mais elles référencent également des ressources transmises sous forme de fichiers.

La transmission des fichiers engendre d'autres problèmes et d'autres failles de sécurité que celles que nous avons détecté dans la partie précédente :

- la taille du fichier transmis ;
- le format du fichier transmis (compatibilité, standard utilisé, etc.) ;
- le contenu du fichier transmis (virus, cheval de Troie, contenu exécutable, etc.).

L'envoi d'un fichier démarre donc par un formulaire dont nous connaissons maintenant la structure. La déclaration du formulaire doit être complétée de l'attribut `enctype="multipart/form-data"` pour assurer le téléchargement du fichier. Un élément `<input type="file" />` recueillera le chemin vers le fichier à télécharger.

```
<body>
  <p>Publier une ressource</p>
  <form id="f1" method="post" action="telechargement.php" enctype="multipart/form-data">
    <fieldset>
      <legend>Dépôt de votre contribution :</legend>
      Entrer la date de publication <input class="question" type="text" id="datePub"
      name="datePub" size="10" /><br /><br />
      Entrer le titre de votre publication <input class="question" type="text" id="titrePub"
      name="titrePub" /><br /><br />
      Ressource à mettre en ligne <input class="question" type="file" id="ressPub"
      name="ressPub" /><br /><br />
      <input name="b1" type="reset" value="effacer" /><input name="b2" type="submit"
      value="Envoyer" />
    </fieldset>
  </form>
</body>
```

Le script *telecharger.php* se charge du traitement (contrôles sur le fichier, puis écriture) :

```
<div class="centre">
<?php
if ( isset($_POST['datePub']) && isset($_POST['titrePub']) && isset($_FILES['ressPub']) ) {
    $datePub = htmlspecialchars($_POST['datePub']);
    $titrePub = htmlspecialchars($_POST['titrePub']);
    if ( $_FILES['ressPub']['error'] == 0 ) {
        // Vérification d'une éventuelle erreur d'envoi
        // Puis traitements liés au fichier transmis
        $tailleFichier = $_FILES['ressPub']['size']; // Récupération de la taille du fichier
        $typeFichier = $_FILES['ressPub']['type']; // Récupération du type du fichier
        $nomFichier = $_FILES['ressPub']['name']; // Récupération du nom complet du fichier
        // Récupération des détails du fichier :
        $detailsFichier = pathinfo($_FILES['ressPub']['name']);
        // Récupération de l'extension du fichier :
        $extensionFichier = $detailsFichier['extension'];
        if ( $tailleFichier <= 10000000 ) { // Vérif. de la taille du fichier (<= 10 Mo)
            // Tableau contenant les extensions autorisées :
            $extensionsPossibles = array('doc', 'docx', 'pdf', 'odt');
            if (in_array($extensionFichier, $extensionsPossibles)) {
                // Vérification de l'extension, elle doit figurer dans la liste autorisée
                echo 'Nous vous remercions d\'avoir mis en ligne la ressource :
                <b>'. $titrePub. '</b>.<br />';
                echo 'Celle-ci sera mise en ligne le : <b>'. $datePub. '</b><br /><br />';
                // Ecriture du fichier téléchargé sur le disque du serveur
                $repertoireDestination = 'files/'; // Choix du dossier de destination
                $resultat = move_uploaded_file($_FILES['ressPub']['tmp_name'],
                $repertoireDestination.$nomFichier); // Ecriture du fichier dans son dossier
                if ( $resultat ) {
                    echo '<h4 class="good">L\'envoi a bien été effectué !</h4>';
                    echo 'Vous pouvez télécharger votre fichier <b>'. $nomFichier. '</b> en
                    cliquant <a href="files/'. $nomFichier. '">ici</a>';
                }
                else {
                    echo '<h4 class="error">Désolé, mais nous n\'avons pas pu télécharger le
                    fichier !</h4>';
                }
            }
            else {
                // L'extension n'est pas autorisée
                echo '<h4 class="error">Désolé, mais l\'extension <b>'. $extensionFichier. '</b>
                du fichier '$_FILES['ressPub']['name']. ' n\'est pas autorisée !</h4>';
            }
        }
        else {
            // La taille du fichier n'est pas autorisée
            echo '<h4 class="error">Désolé, la taille du fichier
            '$_FILES['ressPub']['name']. ' dépasse 10 Mo !</h4>';
        }
    }
    else {
        // Une erreur est survenue au téléchargement
        echo '<h4 class="error">Désolé, mais il y a eu une erreur au téléchargement du
        fichier '$_FILES['ressPub']['name']. ' !</h4>';
    }
}
else {
    echo '<h4 class="error">Erreur, vous n\'avez pas saisi tous les champs !</h4>';
}
?>
</div>
```



**menu :**

- Aller chez google
- Aller à mapage 1
- et puis voilà...

Nous vous remercions d'avoir mis en ligne la ressource : **La vie des informaticiens.**

Celle-ci sera mise en ligne le : **30/01/2032**

**L'envoi a bien été effectué !**

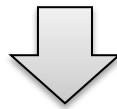
Vous pouvez télécharger votre fichier **laviedesinformaticiens.pdf** en cliquant [ici](#)

Le script se compose de plusieurs parties :

- une vérification de la présence des données du formulaire ;
- une vérification de la composition du fichier (reçu, taille, extension) ;
- l'écriture du fichier s'il a passé tous les contrôles.

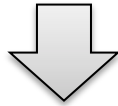
Vérification de la présence des données du formulaire et affectation des variables.

```
if ( isset($_POST['datePub']) && isset($_POST['titrePub']) && isset($_FILES['ressPub']) ) {  
    $datePub = htmlspecialchars($_POST['datePub']);  
    $titrePub = htmlspecialchars($_POST['titrePub']);  
  
    // Suite des traitements  
}  
else {  
    echo '<h4 class="error">Erreur, vous n\'avez pas saisi tous les champs !</h4>';  
}
```



Vérification de la composition du fichier (reçu, taille, extension).

```
if ( $_FILES['ressPub']['error'] == 0 ) {  
    $tailleFichier = $_FILES['ressPub']['size'];  
    $typeFichier = $_FILES['ressPub']['type'];  
    $nomFichier = $_FILES['ressPub']['name'];  
    $detailsFichier = pathinfo($_FILES['ressPub']['name']);  
    $extensionFichier = $detailsFichier['extension'];  
    if ( $tailleFichier <= 10000000 ) {  
        $extensionsPossibles = array('doc', 'docx', 'pdf', 'odt');  
        if (in_array($extensionFichier, $extensionsPossibles)) {  
            // Suite des traitements  
        }  
        else {  
            echo '<h4 class="error">Désolé, mais l\'extension n\'est pas autorisée !</h4>';  
        }  
    }  
    else {  
        echo '<h4 class="error">Désolé, la taille du fichier dépasse 10 Mo !</h4>';  
    }  
} else {  
    echo '<h4 class="error">Désolé, il y a eu une erreur au téléchargement du fichier !</h4>';  
}
```

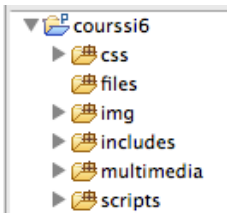


Ecriture du fichier et avertissement au visiteur.

```
echo 'Nous vous remercions d\'avoir mis en ligne la ressource : <b>'.$titrePub.'</b>.<br />';
echo 'Celle-ci sera mise en ligne le : <b>'.$datePub.'</b><br /><br />';
$repertoireDestination = 'files/';
$resultat = move_uploaded_file($_FILES['ressPub']['tmp_name'],
$repertoireDestination.$nomFichier);
if ( $resultat ) {
    echo '<h4 class="good">L\'envoi a bien été effectué !</h4>';
    echo 'Vous pouvez télécharger votre fichier href="files/'.$nomFichier.'">ici</a>';
}
else {
    echo '<h4 class="error">Désolé, mais nous n\'avons pas pu télécharger le fichier !</h4>';
}
```

#### Remarques :

- Le dossier qui reçoit les fichiers téléchargés se nomme *files*. Il est situé à part pour des raisons de sécurité et pour faciliter la gestion des droits sur les dossiers, car il faudra penser à ouvrir les droits d'écriture sur le dossier pour que le script puisse s'exécuter convenablement.
- Il se peut donc que vous ne puissiez pas écrire dans votre dossier si vous n'intervenez pas sur les droits.



On peut compléter la sécurité en changeant les droits sur le fichier téléchargé avec la fonction `chmod()`. Une fois le fichier écrit on peut faire comme suit (les droits attribués sont 0644 soit `rw-r--r--`). Cela veut dire que l'on rend uniquement lisible (`r--`) le répertoire, il n'est accessible en lecture/écriture (`rw-`) qu'à l'administrateur du système.

```
echo 'Nous vous remercions d\'avoir mis en ligne la ressource : <b>'.$titrePub.'</b>.<br />';
echo 'Celle-ci sera mise en ligne le : <b>'.$datePub.'</b><br /><br />';
$repertoireDestination = 'files/';
$resultat = move_uploaded_file($_FILES['ressPub']['tmp_name'],
$repertoireDestination.$nomFichier);
if ( $resultat ) {
    chmod ($repertoireDestination.$nomFichier, 0644); // Changement des droits sur le fichier
    echo '<h4 class="good">L\'envoi a bien été effectué !</h4>';
    echo 'Vous pouvez télécharger votre fichier href="files/'.$nomFichier.'">ici</a>';
}
else {
    echo '<h4 class="error">Désolé, mais nous n\'avons pas pu télécharger le fichier !</h4>';
}
```

L'idéal sera de compléter le script *fonctions.inc.php* en transformant en fonction le contrôle de validité et l'écriture d'un fichier. Ici la fonction se nomme `enregFichier()` et retourne (**return**) la bonne ou mauvaise exécution de l'écriture disque.

```
<?php
/**
 * Fonction de contrôle de validité d'un fichier téléchargé et inscription sur le disque
 *
 * La fonction admet trois paramètres :
 * - $fichier : le fichier en lui-même ;
 * - $tailleMax : la taille maximale du fichier en octets ;
 * - $extensionsPossibles : tableau des extensions autorisées ;
 * - $repertoireDestination : nom du dossier où sera enregistré le fichier (suivi de /).
 */

function enregFichier ($fichier, $tailleMax, $extensionsPossibles, $repertoireDestination) {

    if ( $fichier['error'] == 0 ) {
        // Vérification d'une éventuelle erreur d'envoi
        Traitements liés au fichier transmis

        $tailleFichier = $fichier['size']; // Récupération de la taille du fichier
        $typeFichier = $fichier['type']; // Récupération du type du fichier
        $nomFichier = $fichier['name']; // Récupération du nom complet du fichier
        $detailsFichier = pathinfo($fichier['name']); // Récupération des détails du fichier
        $extensionFichier = $detailsFichier['extension'];

        if ( $tailleFichier <= $tailleMax ) {
            // Vérification de la taille maximum du fichier, ici <= $tailleMax

            if (in_array($extensionFichier, $extensionsPossibles)) {
                // Vérification de l'extension
                $resultat = move_uploaded_file($fichier['tmp_name'],
                    $repertoireDestination.$nomFichier);
                return $resultat; // Retour du résultat de l'écriture du fichier
            }
            else {
                // L'extension n'est pas autorisée
                echo '<h4 class="error">Désolé, mais l\'extension <b>'.$extensionFichier.'</b> du
                    fichier \'$fichier[\'name\']\' n\'est pas autorisée !</h4>';
            }
        }
        else {
            // La taille du fichier n'est pas autorisée
            echo '<h4 class="error">Désolé, la taille du fichier \'$fichier[\'name\']\' dépasse la
                limite autorisée !</h4>';
        }
    }
    else {
        // Une erreur est survenue au téléchargement
        echo '<h4 class="error">Désolé, mais il y a eu une erreur au téléchargement du fichier
            \'$fichier[\'name\']\' !</h4>';
    }
}
?>
```

Le script *telecharger.php* se trouve donc simplifié :

```
<body>
<?php include_once 'includes/fonctions.inc.php'; ?>
<?php include 'includes/entete.inc.php'; ?>
<?php include 'includes/menu.inc.php'; ?>
<div class="centre">
    <?php
        if ( isset($_POST['datePub']) && isset($_POST['titrePub']) && isset($_FILES['ressPub']) )
        {
            $datePub = htmlspecialchars($_POST['datePub']);
            $titrePub = htmlspecialchars($_POST['titrePub']);
            $listeExt = array('doc', 'docx', 'pdf', 'odt');

            if ( enregFichier($_FILES['ressPub'], '10000000', $listeExt, 'files/') ) {
                echo 'Nous vous remercions d\'avoir mis en ligne la ressource :
                <b>'.$titrePub.'</b><br />';
                echo 'Celle-ci sera mise en ligne le : <b>'.$datePub.'</b><br /><br />';
                echo '<h4 class="good">L\'envoi a bien été effectué !</h4>';
            }
            else {
                echo '<h4 class="error">Désolé, mais nous n\'avons pas pu télécharger le fichier
                !</h4>';
            }
        }
        else {
            echo '<h4 class="error">Erreur, vous n\'avez pas saisi tous les champs !</h4>';
        }
    ?>
</div>
</body>
```

Remarque : La fonction devra être complétée de contrôles sur le nom du fichier en lui-même pour éviter les doublons (et donc l'écrasement du fichier précédent) ou sa malformation : nom comportant des espaces, des caractères spéciaux et/ou des caractères accentués.