Chapitre 4

Echanges d'informations entre client et serveur

Le protocole HTTP a défini des possibilités d'envoi d'informations depuis le navigateur vers le serveur. Le serveur retransmet les informations qu'il reçoit aux programmes et scripts qui s'exécutent en parallèle sur le serveur. Deux protocoles précis sont utilisables :

- le mode *get* est le plus ancien et place les valeurs des différentes variables dans la ligne d'URL de la page suivante. Cela représente parfois une contrainte embarassante, notamment pour la transmission des mots de passe.
- le mode *post* est plus discret. En pratique, il veut mieux utiliser ce mode. Les valeurs des variables ne sont pas affichées à l'écran. Notons pourtant qu'elles transitent en clair sur le réseau et qu'un pirate un peu astucieux pourra les lire sans difficulté. Si on veut vraiment disposer d'un échange confidentiel, il faudra avoir recours à un protocole crypté : HTTPS.

En PHP 5, les données récupérées chez le client se trouvent toujours dans un des trois tableaux nommés hyperglobaux : `$_GET`, `$_POST` et `$_REQUEST`, qui s'appliquent respectivement aux modes *get*, *post* et aux deux. L'usage de cette dernière variable est moins sûr puisqu'on ne connaît pas l'origine de la valeur (elle peut également provenir d'un *cookie*).

4.1 Les formulaires

XHTML

Les formulaires jouent un rôle central dans les sites interactifs. Ils permettent à l'utilisateur de remplir des zones de texte, de choisir des éléments dans des listes et de cocher des valeurs. Si l'utilisateur pousse sur le bouton de validation, les valeurs saisies sont renvoyées au serveur http, qui les transmettra à PHP selon le mode choisi

Chaque formulaire comporte les éléments suivants :

- deux tags de début et de fin (il peut donc y avoir plusieurs formulaires sur une page, mais un seul pourra faire l'objet d'un envoi). Le formulaire est délimité par `<form>` et `</form>`. Le premier de ces tags comportent des attributs spécifiant la page à renvoyer lors de la validation (c'est souvent un script) et la méthode choisie pour l'envoi des données.
- au moins un bouton de validation (`type="submit"`) .
- éventuellement un bouton de remise à la valeur initiale des champs (`type="reset"`).
- un ou plusieurs champs de saisie

Voici un exemple de formulaire minimum :

```
<form method="post" action="reponse.php">
   <input type="text" name="nomclient" value="dupont"/>
   <br/>
   <input type="submit" value="envoyer"/>
   <input type="reset" value="recommencer"/>
</form>
```

Lorsqu'on pressera le bouton « Envoyer », la valeur présente dans la zone de texte sera renvoyée au navigateur, en même temps qu'une requête pour afficher la page *reponse.php*.

Voici l'effet du formulaire (placé sur un fond coloré pour le faire ressortir) :

PHP

Si on désire utiliser plusieurs boutons, avec des significations différentes, on peut leur donner à tous le même nom (attribut `name="bouton"`). La valeur lue correspondra à l'attribut `value` du bouton enfoncé par l'internaute.

```
$BoutonEmployé=$_POST["bouton"];
```

4.2 Les zones de texte

XHTML

Une zone de texte doit figurer entre les balises `<form>` d'un formulaire. Elle est formée par le tag autofermant `<input/>`. Il peut avoir plusieurs attributs.

Attributs du tag `<input/>` pour zone de texte	
Attributs	Signification
`type="text"`	Type zone de texte (valeur par défaut)
`type="password"`	Type mot de passe
`type="hidden"`	Type champ caché
`name="nom_champ"`	Nom du champ ou de la variable lors du traitement par le script.
`value="valeur par défaut"`	Valeur affichée au départ et ré-affichée après appui sur un bouton « Reset »
`size="largeur"`	Largeur approximative du champ

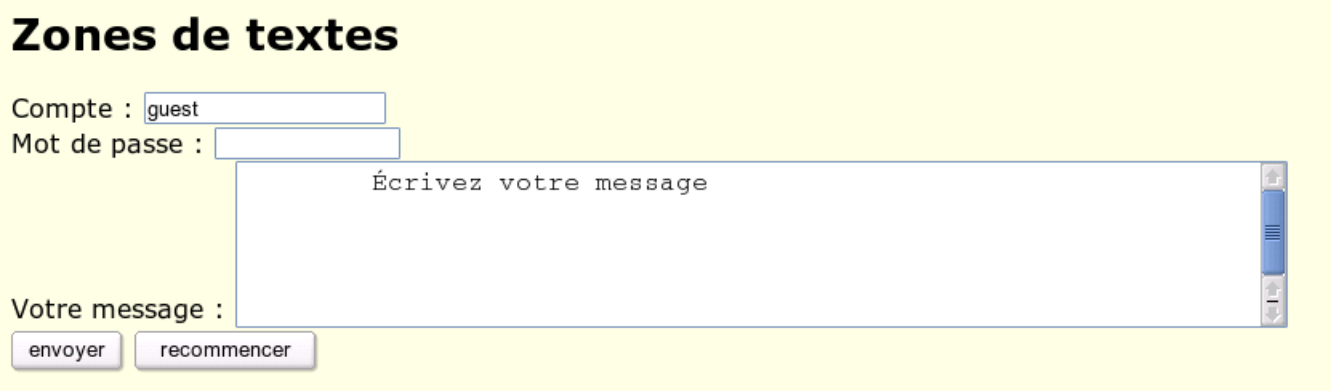
```
<input type="text" name="login" size="40" value="guest"/>
```

Si on désire utiliser une zone où doit figurer un mot de passe, on préféra le type `password`. On évitera ainsi de le voir s'afficher à l'écran. L'attribut `size` est nécessaire pour que la zone de saisie ait une largeur suffisante. Des étoiles apparaîtront lors de la frappe des caractères. Rappelons que la méthode `GET` affichera le mot de passe dans la ligne de saisie du navigateur.

```
<input type="password" name="mdp" size="15"/>
```

Il est également possible de prévoir une zone pour l'écriture d'un message libre :

```
<textarea name="message" rows="5" cols="64" wrap>
    Écrivez votre message
</textarea><br/>
```



On peut placer des champs cachés dans une page HTML. L'utilité de tels champs est de permettre de conserver une valeur à travers plusieurs pages successives [1]. Comme l'utilisateur peut toujours visualiser le code de la page reçue, il ne faut pas espérer lui cacher des informations par ce moyen. L'utilisateur mal intentionné peut aussi recopier la page, modifier les informations cachées avant d'envoyer le contenu du formulaire vers le serveur. Les informations vraiment secrètes doivent demeurer sur le serveur.

```
<input type="hidden" name="transact" value="T501"/>
<input type="text" name="cTexte" value=""/>
```

PHP

En PHP, la page traitant la réponse pourra relire l'élément de clé portant le nom de la zone voulue (exemple : `cTexte`). Cette technique convient pour tous les types de zone.

```
echo 'Vous␣avez␣choisi␣:␣';
echo $_POST['cTexte'];
```

1. Une méthode plus élégante consiste à utiliser des variables de session, que le langage de script permet de conserver sur le serveur. Malheureusement, certains fournisseurs ne proposent pas ce service.

4.3 Les boutons radio

XHTML

Attributs du tag `<input/>` pour un bouton d'option (radio)	
Attributs	Signification
`type="radio"`	Type bouton radio, à spécifier obligatoirement.
`name="nom_champ"`	Nom du champ ou de la variable lors du traitement par le script. Les boutons portant un même nom forment un groupe dans lequel un seul bouton peut être sélectionné.
`value="nom_option"`	Valeur à placer dans la variable lors du renvoi des valeurs
`checked="checked"`	Si cet attribut est présent, le bouton sera présélectionné par défaut ou après appui sur un bouton « Reset »

```
Nombre de PC dans l'entreprise : <BR/>
<input type="radio" name="parc" value="1" checked="checked" /> 1<br/>
<input type="radio" name="parc" value="2" /> 2 à 4<br/>
<input type="radio" name="parc" value="5" /> 5 à 10<br/>
<input type="radio" name="parc" value="10" /> plus de 10<br/>
```

2

PHP

La valeur lue se trouve dans l'élément de clé `parc`.

```
echo 'Vous␣avez␣choisi␣:␣';
echo $_POST['parc'];
```

2. Cette notation un peu étrange est la conséquence du passage à XML: tout attribut doit avoir une valeur mise entre guillemets. Cela semble redondant. Il en est de même pour les anciens attributs sans arguments `selected` et `multiple`.

4.4 Les cases à cocher

XHTML

Attributs du tag <input/> pour une case à cocher	
Attributs	Signification
type="checkbox"	Type case à cocher, à spécifier obligatoirement.
name="nom_champ"	Nom du champ ou de la variable lors du traitement par le script.
checked="checked"	Si cet attribut est présent, la case sera cochée par défaut ou après appui sur un bouton « Reset »

```
Syst&egrave;mes d'exploitation habituellement employ&eacute;:
<input type="checkbox" name="windows" />Windows
<input type="checkbox" name="linux" checked="checked" />Linux
<input type="checkbox" name="macos" />MacOS
```

PHP

Chacune des options cochées crée une variable dans le tableau $_POST. Il suffit de tester l'existence de ces variables pour savoir si la case a été cochée.

```
echo 'Vous avez choisi : ';
if(isset($_POST['windows'])) echo 'Windows';
if(isset($_POST['linux'])) echo 'Linux';
if(isset($_POST['macos'])) echo 'MacOS';
```

4.5 Les listes déroulantes

XHTML

Attributs du tag <select> pour une liste déroulante	
Attributs	Signification
name="nom_champ[]"	Nom du champ ou de la variable lors du traitement par le script. Ce champ est obligatoirement un tableau dont la case 0 contiendra la valeur choisie
size="nombre"	Par défaut, vaut 1 (on affiche une liste déroulante). Si la valeur est supérieure à 1, on affiche une liste de choix.
Attribut du tag <option> pour une liste	
value="valeur"	Valeur renvoyée dans la première case du tableau. En l'absence de cet attribut, c'est tout le texte qui suit le tag qui est renvoyé.
selected="selected"	Effectue la présélection de l'élément ainsi marqué

Il est conseillé d'utiliser l'attribut VALUE dans chaque tag <OPTION>, c'est la valeur de cet attribut qui sera renvoyée. On peut, par exemple, utiliser une valeur de clé primaire dans une table de la base :

```
<select name="Musicien[]">
  <option value="CHO">Chopin</option>
  <option value="MOZ">Mozart</option>
  <option value="MAR">Marin Marais</option>
  <option value="VIV" selected="selected">Antonio Vivaldi</option>
</select>
```

PHP

Si on est certain que l'option multiple n'a pas été employée, il faut lire l'élément 0 de la variable Musicien.

```
echo 'Vous␣avez␣choisi␣:␣';
$Liste=$_POST['Musicien'];
echo $Liste[0];
```

4.6 Les listes à choix multiple

XHTML

La sélection multiple s'adresse à un public d'initiés, car beaucoup d'utilisateurs des environnements graphiques ignorent comment sélectionner plusieurs lignes dans une liste (il suffit de cliquer pendant qu'on enfonce la touche CTRL).

Attributs du tag `<select>` pour une liste à choix multiple	
Attributs	Signification
`multiple="multiple"`	Cet attribut obligatoire permet de distinguer la liste déroulante de la liste à choix multiple.
`name="nom_champ[]"`	Nom du champ ou de la variable lors du traitement par le script. Ce champ est obligatoirement un tableau dont les n premières cases contiendront les valeurs sélectionnées.
`size="nombre"`	Toutes les options sont affichées, mais si elles sont nombreuses, le navigateur peut n'afficher que les quelques premières. L'attribut `SIZE` permet d'exercer un contrôle sur le nombre d'options affichées.
Attribut du tag `<option>` pour une liste	
`value="valeur"`	Valeur renvoyée dans les cases du tableau. En l'absence de cet attribut, c'est tout le texte qui suit le tag qui est renvoyé.
`selected="selected"`	Effectue la présélection de l'élément ainsi marqué

```
<select name="musiciens[]" multiple="multiple">
  <option value="CHO">Chopin </option>
  <option value="MOZ">Mozart </option>
  <option value="MAR">Marais </option>
  <option value="VIV">Vivaldi </option>
</select>
```

PHP

Il faut ici parcourir le tableau récupéré pour obtenir les codes placés dans `OPTION`. Cette technique permet de placer la clé primaire dans la page HTML et de la récupérer directement dans le script, tout en affichant le nom du musicien.

```
echo 'Vous avez choisi les musiciens dont les codes sont : <br/>\n';
$Liste=$_POST['Musiciens'];
foreach($Liste as $Code)
     echo $Code.'<br/>\n';
```

4.7 La sécurité

Après avoir abordé les moyens techniques de récupérer les données depuis un formulaire, notons que la sécurité doit être omniprésente dans la réalisation d'un site. Internet accueille un pourcentage faible de gens mal intentionnés. Malheureusement, ces gens disposent souvent de connaissances pointues et de moyens puissants pour tenter de réaliser leurs coupables desseins. La récupération des données constitue un point sensible de la sécurité d'un site.

Modification du document HTML reçu

La page reçue par la personne qui consulte le site peut facilement être enregistrée sur un disque dur puis modifiée à l'aide d'un simple éditeur de texte. La page « adaptée » peut ensuite être relue par le navigateur, le formulaire rempli et le serveur contacté. C'est dire qu'il ne faut jamais baser la sécurité sur une donnée initialement contenue dans un formulaire envoyé à l'internaute, par exemple dans une variable cachée. Parmi les risques, citons :

- la modification du nombre des champs (ajout ou suppression), de leur nom, de leur type, de leur taille.
- la neutralisation de certains contrôles éventuellement réalisés dans le formulaire au moyen de Javascript.

Il n'existe évidemment aucun moyen d'empêcher l'internaute de procéder à la modification de son formulaire.

Contamination des variables du programme

PHP a longtemps permis la création de variables globales portant le nom des variables contenues dans le formulaire. Cela autorisait le candidat pirate à placer dans son formulaire modifié des variables cachées susceptibles d'influencer le comportement du serveur (par exemple, création d'une variable `$admin` ayant la valeur *vrai*). Il s'agit d'un risque réel qu'on peut facilement combattre :

- ne pas autoriser la création automatique des variables globales : ce comportement est devenu la norme depuis PHP 4.3. PHP 6 supprime complètement ces variables globales. Malheureusement, le paramètre `register_global` est parfois mis à *on* par certains fournisseurs d'accès.
- éviter de placer des valeurs critiques dans des variables globales simples. On peut facilement utiliser des variables de session qui offrent en outre l'avantage de durer au-dela de la simple exécution du script. Exemple :

```
if(condition)
   $_SESSION['admin']=true;
else
   $_SESSION['admin']=false;
...
if($_SESSION['admin']){
   séquence réservée à l'administrateur
```

Placement de données inattendues dans les champs du formulaire

Le pirate peut tenter de placer des caractères spéciaux ou des noms de fichiers critiques dans des variables de formulaire. Une illustration classique de la première technique est l'injection SQL qui consiste à tenter de modifier une requête SQL au moyen d'une donnée tapée par l'utilisateur. Par exemple, comme nom d'utilisateur, on propose la chaîne suivante :

```
Dupont' --
```

On voit clairement que si la requête envoyée au SGBD est du genre

```
SELECT id FROM Users WHERE nom = '$user' AND pwd = '$pwd';
```

la requête exécutée sera

```
SELECT id FROM Users
WHERE nom = 'Dupont' -- ' AND pwd = '$pwd';
```

qui permettra de s'identifier comme Dupont, sans mot de passe. Notons que le pirate aurait pu faire plus de mal en donnant comme nom :

```
Dupont'; DROP TABLE Users;
```

Un autre cas classique consiste à récupérer une donnée de l'utilisateur comme nom de fichier à afficher. Que va-t-il se passer si l'utilisateur propose `/etc/passwd` ?

Les réactions à ces dangers sont multiples :

– ne jamais utiliser une donnée fournie par l'utilisateur sans vérifier son contenu. On peut éliminer les risques au moyen de la fonction `addslashes()` qui « échappe » les délimiteurs de chaînes, ou encore `mysql_real_escape_string()` qui élimine tous les caractères « anormaux ». On peut aussi passer la chaîne au tamis d'une expression régulière. Par exemple, voici comment vérifier qu'un mot de passe ne contient que des caractères alphanumériques :

```
$pwd=ereg_replace('[^[:alnum:]]','A',$pwd);
```

Tout caractère non réglementaire sera remplacé par A.

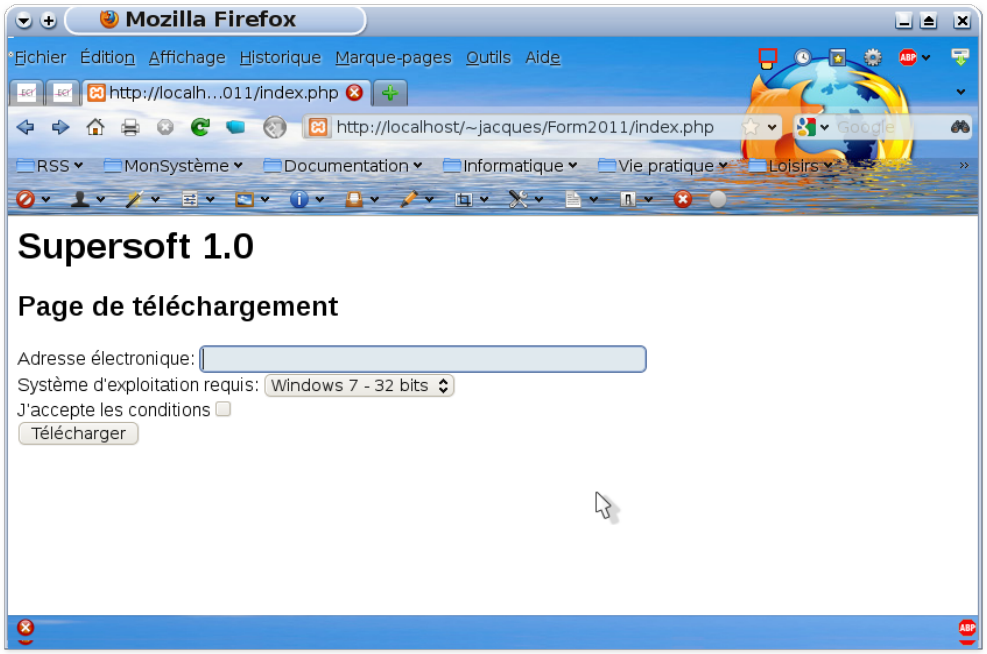
– s'il faut choisir dans une liste, bien que la liste soit fournie, elle peut avoir été altérée. On vérifiera, par exemple au moyen de `switch`, que la donnée renvoyée est bien une de celles qui étaient proposées.

Règle finale

On pourra énoncer une règle simple à mettre en pratique dans toutes les interactions avec un internaute :

NE JAMAIS FAIRE CONFIANCE À UNE DONNÉE FOURNIE PAR UN UTILISTEUR.

4.8 Exemple concret



Je vais donner un exemple concret d'une page comportant un formulaire. Voici le comportement désiré :

- un formulaire vierge est proposé lors du premier affichage (avec éventuellement certaines zones remplie avec une valeur par défaut). Nous allons y placer un champ pour lire l'adresse électronique de l'utilisateur, une liste déroulante pour lui permettre de choisir son système d'exploitation et une case à cocher pour accepter nos conditions.
- après une première validation, les données proposées par l'utilisateur sont replacées dans le formulaire, qu'il peut ainsi modifier selon ses désirs
- si la case à cocher est sélectionnée et que le champ d'adresse contient une adresse électronique valide, on procède à l'affichage de la suite (à savoir une réponse à la requête de l'utilisateur). Le formulaire n'est plus réaffiché.

Concrètement, la page comportera trois parties :

1. une partie d'analyse de la réponse de l'utilisateur. Notons que cette partie n'a pas de sens lors du premier affichage. Il faut détecter cette circonstance et ne rien faire.
2. afficher (si nécessaire) le formulaire avec des données vierges ou partiellement remplies par l'utilisateur.
3. afficher la réponse à la requête.

En fait les deuxième et troisième parties, dans notre optique, s'excluent mutuellement.

4.8.1 Analyse des données de l'utilisateur

La question se pose de savoir si l'utilisateur a déjà vu et commencé à remplir le formulaire. Pour cela, il suffit de placer un champ caché dans le formulaire. Si ce champ renvoie une valeur, c'est que le formulaire a déjà été lu [3]. Pour le reste, nous allons initialiser les variables `$email`, `$os` et `$conditions`.

3. La technique du champ caché ne pose pas de problème de sécurité dans ce cas. Si l'utilisateur arrive à simuler l'existence d'une valeur pour ce champ caché, c'est qu'il a déjà vu notre formulaire, ce qui est précisément ce que nous tentons de déterminer.

```
$formulaireRempli = false;
/* -- 1 -- Analyse des résultats */
// Si pas de réponse de l'utilisateur
if (!isset($_POST['Lu'])) {
     // les réponses sont nulles
     $email=$os=$conditions=null;
}
else{
     // Lire les réponses
     $email = lireValeur('email');
     $conditions = lireValeur('conditions');
     $os = lireValeur('os'); // toujours initialisé (défaut)
     /* Expression rationnelle:
      * des caractères initiaux
      * arobas
      * domaine point 2à6 lettres
      */
      $Validation='#^[\w.-]+@[\w.-]+\.[a-zA-Z]{2,6}$#';
      if ($conditions and preg_match($Validation, $email)) {
            $formulaireRempli = true;
      }
}
```

Dans le but de ne pas alourdir la lecture des champs du formulaire, j'utilise un fonction qu'on peut définir en début de page :

```
function lireValeur($Champ) {
    // pas de valeur : renvoyer null
    if (!isset($_POST[$Champ])) {
        return null;
    } else {
        $variable = $_POST[$Champ];
        // si la variable est un tableau (provient d'une liste)
        // lire la première valeur
        if (is_array($variable)) {
            return $variable[0];
        }
        // autrement renvoyer la valeur lue
        else {
            return $variable;
        }
    }
}
```

Si la variable n'existe pas, on renvoie `null`. Autrement, on teste si la variable est un tableau (cas d'une liste). Dans ce cas, on considère qu'une seule valeur peut être renvoyée à la fois, on prend alors la première valeur.

Pour la suite, il paraît plus simple d'utiliser un tableau pour représenter les différents systèmes d'exploitation. A placer également en début de page.

```
$systemes = array(
    'W7'=>'Windows␣7␣-␣32␣bits',
    'W7-64'=>'Windows␣7␣-␣64␣bits',
```

```
    'U1010'=>'Ubuntu␣10.10',
    'OS114'=>'OpenSuse␣11.4',
    'Mac'=>'Mac␣OSX',
    'Deb6'=>'Debian␣Squeeze',
    'Other'=>'Autre␣système'
);
```

4.8.2 Affichage du formulaire

La liste déroulante se remplit avec une boucle.

```
if (!$formulaireRempli):
     /* -- 2 -- Formulaire */
     ?>
     <form action="index.php" method="POST">
          <input type="hidden" name="Lu" value="" />
          Adresse électronique:
        <input value="<?= $email ?>" type="text" name="email" size="40" />
        <br/>
        Système d'exploitation requis:
        <select name="os">
            <? foreach($systemes as $cle=>$nom):?>
               <option value="<?= $cle?>"
                  <?if($os==$cle):?>selected="selected"<?endif?>>
                  <?=$nom?>
               </option>
            <? endforeach;?>
        </select><br/>
        J'accepte les conditions
        <input type="checkbox" name="conditions"
             <?if($conditions):?>checked=""checked"<?endif?> />
        <br/>
        <input type="submit" value="Télécharger" name="submit" />
     </form>
<?php
else:
```

4.8.3 Exploitation du formulaire

```
else:
     /* -- 3 -- Réaction du serveur */
?>
Vous avez demandé la version <?= $systemes[$os]?> de notre logiciel.
<br/>
<a href="http://supersoft.com/download/supersoft1.0-<?=␣$os?>.zip">
   Téléchargement
</a>
<?endif;?>
```

4.8.4 Listing complet de la page

```
<?php

function lireValeur($Champ) {
    // pas de valeur : renvoyer null
    if (!isset($_POST[$Champ])) {
        return null;
    } else {
        $variable = $_POST[$Champ];
        // si la variable est un tableau (provient d'une liste)
        // lire la première valeur
        if (is_array($variable)) {
            return $variable[0];
        }
        // autrement renvoyer la valeur lue
        else {
            return $variable;
        }
    }
}

$systemes = array(
    'W7'=>'Windows 7 - 32 bits',
    'W7-64'=>'Windows 7 - 64 bits',
    'U1010'=>'Ubuntu 10.10',
    'OS114'=>'OpenSuse 11.4',
    'Mac'=>'Mac OSX',
    'Deb6'=>'Debian Squeeze',
    'Other'=>'Autre système'
);

?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title></title>
    </head>
    <body>
        <h1>Supersoft 1.0</h1>
        <h2>Page de téléchargement</h2>
        <?php
        $formulaireRempli = false;
        /* -- 1 -- Analyse des résultats */
        // Si pas de réponse de l'utilisateur
        if (!isset($_POST['Lu'])) {
            // les réponses sont nulles
            $email=$os=$conditions=null;
        }
        else{
            // Lire les réponses
            $email = lireValeur('email');
```

```
            $conditions = lireValeur('conditions');
            $os = lireValeur('os'); // toujours initialisé (défaut)
            /* Expression rationnelle:
             * des caractères initiaux
             * arobas
             * domaine point 2à6 lettres
             */
            $Validation='#^[\w.-]+@[\w.-]+\.[a-zA-Z]{2,6}$#';
            if ($conditions and preg_match($Validation, $email)) {
                $formulaireRempli = true;
            }
        }
        if (!$formulaireRempli):
            /* -- 2 -- Formulaire */
        ?>
            <form action="index.php" method="POST">
                <input type="hidden" name="Lu" value="" />
                Adresse électronique:
                <input value="<?= $email ?>" type="text" name="email" size="40" />
                <br/>
                Système d'exploitation requis:
                <select name="os">
                    <? foreach($systemes as $cle=>$nom):?>
                    <option value="<?= $cle?>"
                        <?if($os==$cle):?>selected="selected"<?endif?>><?=$nom?>
                    </option>
                    <? endforeach;?>
                </select><br/>
                J'accepte les conditions
                <input type="checkbox" name="conditions"
                   <?if($conditions):?>checked=""checked"<?endif?> />
                <br/>
                <input type="submit" value="Télécharger" name="submit" />
            </form>
        <?php
            else:
            /* -- 3 -- Réaction du serveur */
                ?>
            Vous avez demandé la version <?= $systemes[$os]?> de notre logiciel.
            <br/>
            <a href="http://supersoft.com/download/supersoft1.0-<?= $os?>.zip">
                Téléchargement
            </a>
            <?endif;?>
    </body>
</html>
```