

Transmission de données

Dr N. BAME

Introduction

- Transmission de données entre les différentes pages d'un site web.
- Trois modes de transmission
 - ✓ Par des URL
 - ✓ Par formulaires
 - ✓ Par l'intermédiaire de fichiers : écriture/lecture

Transmettre des données avec l'URL

- URL : Unified Resource Locator
 - Représenter une adresse sur le web
- Possibilité de transmission de données entre deux pages grâce aux URL
 - Envoi de paramètres dans l'URL

Transmettre des données avec l'URL

- Fonctionnement

- Considérons un site *www.monsite.com* et une page *page1.php* du site.

- **Envoi de paramètres** dans l'URL:

- *www.monsite.com/page1.php?param1=V1¶m2=V2&pa---*
¶mN=Vn

- Exemple

- *www.monsite.com/page1.php?classe=L3Pro&Libelle=Licence3Pro&NBEtudiant=12*

- On peut écrire autant de paramètres que l'on veut à condition que
l'URL ne dépasse pas 255 caractères

- Remarque

- La transmission de données par les URL pose un réel problème de sécurité dans la mesure où les données transmises peuvent être modifiées à travers l'URL

Transmettre des données avec l'URL

- Récupérer les paramètres dans la page de réception
 - ✓ Utilisation de la fonction `$_GET['parami']` pour récupérer la valeur qui a été associée au paramètre *parami*.
 - ✓ Ainsi on peut manipuler dans la page les valeurs transmises
 - ✓ Exemple

```
$classe=$_GET['classe'];  
echo $classe;
```

Contrôler la valeur des paramètres

- Tester la présence d'un paramètre avec la fonction **isset()**

```
if(isset($_GET['param']))
```

- Exemple

```
<?php
```

```
    if (isset($_GET['prenom']) AND isset($_GET['nom']) )
```

```
    {
```

```
        Traitements
```

```
    }
```

```
?>
```

Transmission de données avec les formulaires

- **Les formulaires**

- Interactivité

- Saisie de données dans une base de données
 - Affichage de données dynamiques et personnalisées

- Contrôle dynamique

- Contrôle de saisie

Création de formulaires

- La création de formulaire se fait en HTML à l'aide la balise paire `<form>`.

```
<form method="post" action="traitement.php">
```

```
<p>
```

On insèrera ici les éléments de notre formulaire.

```
</p>
```

```
</form>
```

- Remarque** : Les attributs *method* et *action* sont indispensables pour manipuler les formulaires

Les formulaires : la méthode

- plusieurs moyens «méthodes» pour envoyer des données du formulaire
- **GET :**
 - les données transiteront par l'URL. On pourra les récupérer grâce à l'array `$_GET`.
 - Cette méthode est assez peu utilisée car on ne peut pas envoyer beaucoup d'informations dans l'URL
- **POST:**
 - permet d'envoyer autant de données que l'on veut, ce qui fait qu'on la privilégie le plus souvent.
 - On pourra récupérer les données grâce à l'array `$_POST`
 - *method='POST'*


Les formulaires : la cible

- La cible permet de définir la page à appeler à la validation du formulaire
- Elle est définie par l'attribut *action*
- Cette page recevra les données du formulaire et sera chargée de les traiter.
- Le nom du fichier cible est défini avec *l'attribut action*
- Si le fichier contenant le formulaire est lui-même la cible, on peut utiliser la variable `$_SERVER["PHP_SELF"]`, qui contient le nom du fichier en cours d'exécution comme valeur de l'attribut
 - Pas la nécessité de modifier la valeur de l'attribut *action* en cas de maintenance du code ayant entraîné le changement du nom du fichier
 - `<form method="post" action="<? echo $_SERVER['PHP_SELF'] ?>"`

Éléments de formulaire : input

- Permet de créer les composants classiques des formulaires
- Avec des aspects et des rôles très différents
- L'attribut *name* est obligatoire, car c'est lui qui permet d'identifier les champs côté serveur et ainsi de récupérer leur contenu.
- La différenciation de ces composants s'effectue simplement en définissant la valeur de leurs attributs, et de l'attribut *type*.
- Plusieurs types existent
 - "text", "password", "checkbox", "radio", "submit", "reset", "file" et "hidden", ...
- Généralement un libellé est associé à une zone de texte à l'aide de la balise paire `<label>`
- Par la suite on doit lier le libellé à la zone de texte associée. Pour cela, l'attribut *for* est utilisé dans le label avec une valeur correspondant au name de la zone

L'élément `<input type="text" />`

- Les petites zones de texte 
 - crée un champ de saisie de texte d'une seule ligne
- Quelques attributs supplémentaires
 - ✓ `size="nombre"` : taille de la zone en nombre de caractères.
 - ✓ `maxlength="nombre"` : nombre maximal de caractères que l'utilisateur est autorisé à saisir.
 - ✓ `value="texte"` : texte par défaut tant que l'utilisateur ne l'a pas modifié.
 - C'est cette valeur qui est transmise au serveur si l'internaute ne saisit aucun texte
 - ✓ `placeholder` : donne une indication sur le contenu du champ et disparaît dès que le visiteur clique à l'intérieur du champ

```
<input type="text" name="ville" size="30" maxlength="40" value="Dakar"/>
```

L'élément `<input type="text" />`

L'attribut value

- Pour des raisons d'ergonomie, il est préférable que le **texte par défaut** défini à l'aide de l'attribut value **s'efface tout seul au moment où l'utilisateur clique dessus** car cela lui évite de devoir l'effacer.
- Pour cela, il suffit d'utiliser une instruction **JavaScript** très simple.

- pour réagir à l'événement clic :

```
<input type="text" name="ville" size="30" maxlength="40" value="Votre ville" onclick="this.value="" />
```

- Pour que le texte s'efface dès que la zone de texte reçoit le focus (le curseur passe dans la zone) :

```
<input type="text" name="ville" size="30" maxlength="40" value="Votre ville" onfocus="this.value="" />
```

L'élément `<input type="password" />`

- Crée un champ de saisie de *mot de passe* semblable à un champ texte mais dans lequel les caractères *saisis sont invisibles et remplacés par des astérisques*.
- Les attributs *size* et *maxlength* y jouent le même rôle que précédemment.

```
<input type="password" name="code" size=" 10"
maxlength="6"/>
```

L'élément `<input type="checkbox" />`

Crée une case à cocher utilisée pour effectuer un ou plusieurs choix parmi ceux qui sont pré-établis par le programmeur.

L'attribut ***value*** contient le texte **qui sera transmis au serveur** si l'utilisateur **coche la case**. Il est **obligatoire**.

Dans le cas de groupes de cases à cocher, il faut que

- ✓ le **nom** de tous les composants **soit le même** et
- ✓ qu'il soit **suivi de crochets ouvrants et fermants** de façon à récupérer les valeurs dans un tableau

Exemple

```
<input type ="checkbox" name="lang[]" value="français" />
```

```
<input type ="checkbox" name="lang[]" value="anglais" />
```

L'élément `<input type="radio" />`

- Créer un bouton radio.
- Employé **seul**, un bouton radio peut servir à valider **des choix**.
- Employé **en groupe**, il implique, à la différence de cases à cocher, **qu'un seul choix est autorisé**.
 - Dans ce cas, **tous les boutons** radio du groupe **doivent avoir une même valeur pour l'attribut "name"**. Le fait d'en activer un, désactive celui qui l'était auparavant.
- L'attribut **`checked="checked"`** définit le bouton coché par défaut.
- L'attribut **`value`** joue le même rôle que pour les cases à cocher et est également indispensable.

```
<label>Débutant</label><input type="radio" name="capa" value="débutant" />
```

```
<label>Initié</label><input type="radio" name="capa" value="initié"/>
```


L'élément `<input type="submit" />`

- Crée un bouton sur lequel l'utilisateur doit cliquer pour déclencher l'envoi des données de tout le formulaire vers le serveur.
- Il est indispensable d'avoir au moins un bouton d'envoi par formulaire, mais il est possible d'en utiliser plusieurs.
 - Le clic sur l'un de ces boutons est alors analysé par le script désigné par l'attribut *action* de l'élément `<form>`.
 - Cela permet d'effectuer des tâches spécialisées en fonction de la valeur associée à chaque bouton grâce à son attribut *value*. C'est le contenu de l'attribut *value* qui constitue le texte visible du bouton dans le formulaire.

`<input type="submit" value="Envoyer" />`

L'élément `<input type="reset" />`

- Crée un bouton de réinitialisation du formulaire et **non d'effacement systématique**, comme on le croit souvent.
- Si les éléments du formulaire ont des attributs value qui définissent des valeurs par défaut, ce sont ces valeurs qui apparaissent au démarrage de la page et qui sont réaffichées si l'utilisateur clique sur le bouton **reset**.
- Le contenu de l'attribut value du bouton d'effacement constitue le texte visible du bouton dans le formulaire.
- **Exemple**
`<input type="reset" value="Annuler" />`

L'élément `<input type="file" />`

- Permet le transfert de fichiers du poste client vers le serveur.
- Crée un **champ de saisie** de même aspect qu'un champ de texte et **un bouton de sélection de fichier** permettant de choisir le fichier à transférer.
- L'attribut ***name*** est obligatoire.
- On peut utiliser en plus les attributs :
 - ***size*** limitant la taille de la zone de saisie,
 - et plus particulièrement l'attribut ***accept***, qui définit le ou les types de fichier acceptés en transfert.
 - Pour sécurité, l'utilisation de cet attribut est recommandée car il ***permet de limiter le transfert à certains types de fichiers*** bien précis et de refuser les autres.
- **Exemple**

```
<input type="file" name="monfichier"  
accept="image/gif,image/jpeg" size="30"/>
```

L'élément `<input type="hidden" />`

- Crée un champ caché n'ayant, comme son nom l'indique, **aucun rendu visuel** dans le formulaire **mais permettant de transmettre des informations invisibles** pour l'utilisateur.
- Les informations sont contenues dans l'attribut ***value***. L'attribut ***name est obligatoire***.
- **Exemple**
`<input type="hidden" name="MAX_FILE_SIZE" value="20000"/>`

Nouveaux types en HTML5

- HTML 5 introduit de nouveaux types de champs de formulaires. Parmi ceux-ci :
`email, url, tel, number, color, range, ...`
- Ces différents types viennent s'ajouter aux valeurs classiques de HTML que nous venons de voir

HTML5 : l'élément <input .../>

type= "email"

- Le champ semble *a priori* identique
 - mais le navigateur sait désormais que l'utilisateur doit saisir une adresse e-mail.
 - Il peut afficher une indication si l'adresse n'est pas un email

type= "url"

- le champ accueille des URL absolues. Pour laisser un
- lien vers son site Internet

type= "tel"

- le champ est destiné aux numéros de téléphone.

type= "range"

- le champ permet de sélectionner un nombre avec un curseur (aussi appelé *slider*)
 - On peut utiliser les attributs **min**, **max** et **step** pour restreindre les valeurs disponibles.



L'élément <input .../>

type= "number"

- le champ permet de sélectionner un nombre avec
 - ✓ On peut utiliser les attributs `min`, `max` et `step` pour restreindre les valeurs disponibles.
 - ✓ champ s'affiche en général avec des petites flèches pour changer la valeur

type= "color"

- le champ permet de saisir une couleur en ouvrant automatiquement une boîte à couleurs.

type= "search"

- Afficher un champ de recherche

L'élément <input .../>

type= "date"

- le champ permet de saisir une date au format jj/mm/yyyy (15/02/1992 par exemple) ;

type= "time"

- pour l'heure (13:37 par exemple) ;

type= "week"

- Pour le jour de la semaine

type= "month"

- pour le mois

type= "datetime"

- pour la date et l'heure (avec gestion du décalage horaire) ;

type= "datetime-local "

- pour la date et l'heure (sans gestion du décalage horaire).

Example

```
<form method="post" action="traitement.php">
  <p>
    <label for="pseudo "> Nom d'utilisateur : </label>
    <input type="text" name="pseudo" id="pseudo"/>
  </p>
  <p>
    <label for=" pwd ">Mot de passe : </label>
    <input type="password" name="pwd" id=" pwd"/>
  </p>
  <p>
    <input type="submit" value="Se connecter" />
  </p>
</form>
```

L'élément `<textarea>`

- Crée un champ de saisie de texte permettant la saisie sur plusieurs lignes.
- Cet élément comporte une balise de fermeture `</textarea>` et un contenu de texte par défaut affiché dans la zone de texte.
- Les attributs **cols** et **rows**, qui donnent respectivement le nombre de colonnes et de lignes de la zone de texte, doivent être définis.
- **Exemple**
`<textarea name="commentaire" cols="45" rows="8" >`
 Tapez vos commentaires ici
`</textarea>`

L'élément `<select>`

- Crée une liste de sélection d'options parmi lesquelles l'utilisateur fait un choix, chaque option devant être définie par un élément `<option>` séparé.
- L'élément `<select>` a la structure suivante :
`<select name="maliste">`
 `<option value="valeur 1"> Texte choix 1</option>`
 `<option value="valeur 2"> Texte choix 2</option>`
 `<option value="valeur 3"> Texte choix 3</option>`
`</select>`

L'élément optgroup

- On peut grouper les options des listes déroulantes sous un dénominateur commun grâce à l'élément `optgroup` et son attribut `label`

```
<select name="departement">
  <optgroup label=Dakar>
    <option value= "Pikine" >Pikine<option>
    <option value= "Rufisque" > Rufisque <option>
    <option value= "Guédiawaye" > Guédiawaye<option>
  </optgroup>
  <optgroup label="Thies">
    <option value= " Thies " >Thies <option>
    <option value= " Tivaoune " > Tivaoune <option>
    <option value= " Mbour " > Mbour <option>
  </optgroup>
</select>
```

L'élément <select>

- comporte les attributs suivants :
 - name="nom_select"*, Obligatoire, donne le nom de la liste.
 - size="Nombre"*, Détermine le nombre de choix visibles simultanément. Par défaut, sa valeur est 1.
 - multiple="multiple"*, Autorise la sélection de plusieurs options simultanément.
- L'élément *<option>* comporte les attributs suivants :
 - value="chaine"*, obligatoire.
 - Définit la valeur transmise au serveur si l'option est sélectionnée.
 - selected="selected"* définit l'option qui est sélectionnée par défaut dans la liste si l'utilisateur ne fait pas de choix.

Les attributs placeholder et required

- L'attribut **placeholder**, sert à donner davantage d'indications sur les champs aux visiteurs.
 - Il sert à donner un exemple de remplissage de champ.
- Pour rendre une question du formulaire **obligatoire**, on doit utiliser l'attribut **required**.

Les éléments fieldset et legend.

- L'élément **fieldset** sert à organiser le formulaire en différentes rubriques.
 - Cela peut être utile si celui-ci devient long.
- On doit ensuite donner une légende à chaque rubrique grâce à l'élément **legend**.

```
<form>
```

```
  <fieldset>
```

```
    <legend>Etat Civil</legend>
```

```
    ...
```

```
  </fieldset>
```

```
  <fieldset>
```

```
    <legend>Formations et diplômes</legend>
```

```
    ...
```

```
  </fieldset>
```

```
</form>
```

Récupération des données du formulaire

- Après la validation du formulaire, une requête HTTP est envoyée au serveur à destination du script désigné par l'attribut action de l'élément **<form>**.
- La requête contient **toutes les associations** entre les **noms** des **champs** et leur **valeurs**.
- Ces associations se trouvent dans **l'en-tête HTTP** si la méthode **POST** est utilisée et **dans l'URL** s'il s'agit de la méthode **GET**.

Récupération des données du formulaire

Valeurs uniques

- Les valeurs uniques proviennent des champs de formulaire dans lesquels l'utilisateur ne peut entrer qu'une valeur, **un texte** par exemple, ou ne peut faire **qu'un seul choix**
 - bouton radio, liste de sélection à choix unique, ...
- Ces valeurs sont contenues sur le serveur dans des tableaux associatifs dits superglobaux appelés ***\$_POST*** et ***\$_GET***, selon la méthode choisie.
 - Les clés de ces tableaux sont les noms associés aux champs par l'attribut *name*.

Exemple

```
<?php
if(isset($_POST["nom"]) && isset($_POST["prenom"]))
{
    echo "<h2> Bonjour ". $_POST["prenom"]. " ". $_POST["nom"] . "</h2>"
}
?>
```

Récupération des données du formulaire

Valeurs uniques

- L'exemple contrôle d'abord l'existence des variables `$_POST["nom"]` et `$_POST["prenom"]`,
 - ✓ qui représentent respectivement le texte saisi et la valeur associée à la case cochée de façon à n'afficher le message qu'après l'envoi des données.
- Ces variables n'existent que si l'utilisateur a complété les champs et a cliqué sur le bouton d'envoi.
- Lorsqu'elles existent, elles sont utilisées pour créer un affichage de bienvenue

Récupération des données du formulaire

Les valeurs multiples

Certains champs de formulaire peuvent permettre aux visiteurs de saisir plusieurs valeurs sous un même nom de composant.

- Cela peut concerner un ***groupe de cases à cocher*** ayant le même attribut ***name***,
 - par exemple, dont il est possible de cocher une ou plusieurs cases simultanément.
- Ceci peut également être le cas d'une ***liste de sélection*** ayant toujours un ***nom unique*** mais dans laquelle l'attribut ***multiple="multiple"*** est défini.
- Il est enfin possible de donner le même nom à des éléments de saisie de texte différents, mais cela présente moins d'intérêt.

Récupération des données du formulaire

Les valeurs multiples

- Dans tous les cas, ce n'est pas une valeur scalaire mais un tableau qui est récupéré côté serveur.
- Il faut pour cela faire suivre le nom du composant de crochets, comme pour créer une variable de type array.
Bleu:<input type="checkbox" name="choix[]" value="bleu" />
Blanc:<input type="checkbox" name="choix[]" value="blanc" />
- l'utilisateur peut cocher les deux cases simultanément.
- On récupère ces valeurs dans les variables suivantes :
\$_POST["choix"][0] qui contient la valeur "bleu"
\$_POST["choix"][1] qui contient la valeur "blanc"

Maintien de l'état du formulaire

- Lorsque le script contenant le formulaire est chargé du traitement des données, l'ensemble de la page est réaffiché après traitement, de même que l'ensemble du formulaire.
- Le formulaire se retrouve alors dans son état initial, et toutes les saisies effectuées sont effacées
- Pour la zone de saisie de texte dont l'attribut ***name*** a la valeur "***nom***", il suffit de définir l'attribut ***value*** avec la variable ***\$_POST["nom"]***
 - **non** sans avoir au préalable **contrôlé l'existence de cette variable**.
 - Lors du premier affichage de la page, la zone est donc vide ou contient le dernier nom saisi.

```
<?php if(isset($_POST["nom"])) echo $_POST["nom"]?>
```

Maintien de l'état du formulaire

- Pour les boutons radio dont l'attribut ***name*** a la valeur "***niveau***" et qui permettent le choix entre les valeurs "***Débutant***" et "***Initié***", il faut définir l'attribut ***checked*** du bouton choisi à la valeur "***checked***" en fonction de la valeur de la variable `$_POST["niveau"]`

```
<?php
```

```
if(isset($_POST["niveau"]) && $_POST["niveau"]=="débutant")  
    echo "checked='checked'"
```

```
?>
```

Lecture et Ecriture dans un fichier

- Enregistrement de fichier sur le disque dur
- Création/modification/suppression de fichier
- Ecriture et lecture de données d'un fichier
- Nécessité d'autorisation d'écriture dans un dossier du serveur sous linux avec la commande `chmod`

Fonctions de gestions de fichiers

fopen(\$file [, \$mode]) : ouverture du fichier identifié par son nom **\$file** et dans un mode **\$mode** particulier, retourne un identificateur **\$fp** de fichier ou **FALSE** si échec

fclose(\$fp) : ferme le fichier identifié par le **\$fp**

feof(\$fp) : teste la **fin** du fichier

file_exists(\$file) : indique si le fichier **\$file** **existe**

filesize(\$file) : retourne la **taille** du fichier **\$file**

touch(\$file) : pour créer un fichier vide

Fonctions de gestions de fichiers

filetype(\$file) : retourne le **type** du fichier **\$file**

unlink(\$file) : détruit le fichier **\$file**

copy(\$source, \$dest) : **copie** le fichier **\$source** vers **\$dest**

rename(\$old, \$new) **renomme** : le fichier **\$old** en **\$new**

fseek(\$fp, position): place le curseur à la position voulue

Fonctions de gestions de fichiers

- ***fgets(\$fp, \$length)*** : lit une ligne de \$length caractères au maximum
- ***fgetc(\$fp)*** : **lit** un caractère
- ***readfile(\$file)*** : **affiche** le fichier **\$file**
- **fread()** : permet de lire plusieurs octets en une fois et retourne une chaîne de caractères.
- **file_get_contents()** : permet de lire entièrement le contenu d'un fichier (en une passe).
 - Elle prend en argument l'adresse du fichier et retourne une chaîne de caractères avec l'intégralité du contenu.

Fonctions de gestions de fichiers

- ***fputs(\$fp, \$str)*** : écrit la chaîne **\$str** dans le fichier identifié par \$fp
- ***fwrite()*** : fonction d'écriture simple : Elle prend en paramètre le descripteur de fichier et la chaîne à insérer dans le fichier.
- ***file_put_contents()***: Ecrire entièrement le contenu d'un fichier.

Modes d'ouverture d'un fichier

- **r** : Ouvre le fichier en **lecture seule**. Cela signifie que vous pourrez seulement lire le fichier.
- **r+** : Ouvre le fichier en **lecture et écriture**. Vous pourrez non seulement lire le fichier, mais aussi y écrire.
- **a** : Ouvre le fichier en **écriture seule**. Mais il y a un avantage : si le fichier **n'existe pas**, il est **automatiquement créé**.
- **a+** : Ouvre le fichier en **lecture et écriture**. Si le fichier n'existe pas, il est **créé automatiquement**.
 - Attention : le répertoire doit avoir un CHMOD à 777 dans ce cas !
 - A noter que si le fichier existe déjà, le texte sera rajouté à la fin.
- **w**: **Création** et **écriture seule**
- **W+**: **Création** et **écriture/lecture**

Exemple : affichage des lignes d'un fichier

<?php

```
$file = "fichier.txt" ;  
if($fd = fopen($file, "r"))    // ouverture du fichier en lecture  
{  
    // tant que la fin de fichier n'est pas atteinte  
    while(!feof($fd))  
    {  
        $str = fgets($fd, 1024); /* lecture jusqu'à fin de ligne où des 1024  
                                   premiers caractères */  
        echo $str;           // affichage  
    }  
    fclose ($fd);           // fermeture du fichier  
}  
else  
    die("Ouverture du fichier <b>$file</b> impossible.");
```

?>

Exemple de manipulation de fichier

```
<?php
$path='mes_fichiers';
$f1='fichier1.text';
$file=$path.'/'.$f1;
echo $file;
$fp=fopen($file, "w+");
$i=1;
while($i<11)
{
    if($fp)
    {
        fputs($fp,$i."==>Test avec le premier fichier\n");
        $i++;
    }
}
fclose($fp);
$fp=fopen($file, "r");
if($fp)
{
    echo '<br/>';
    while(!feof($fp))
    {
        $line=fgets($fp);
        echo $line.'<br/>';
    }
}
```