

Programmation web II



Mohamed Karim Abdmouleh

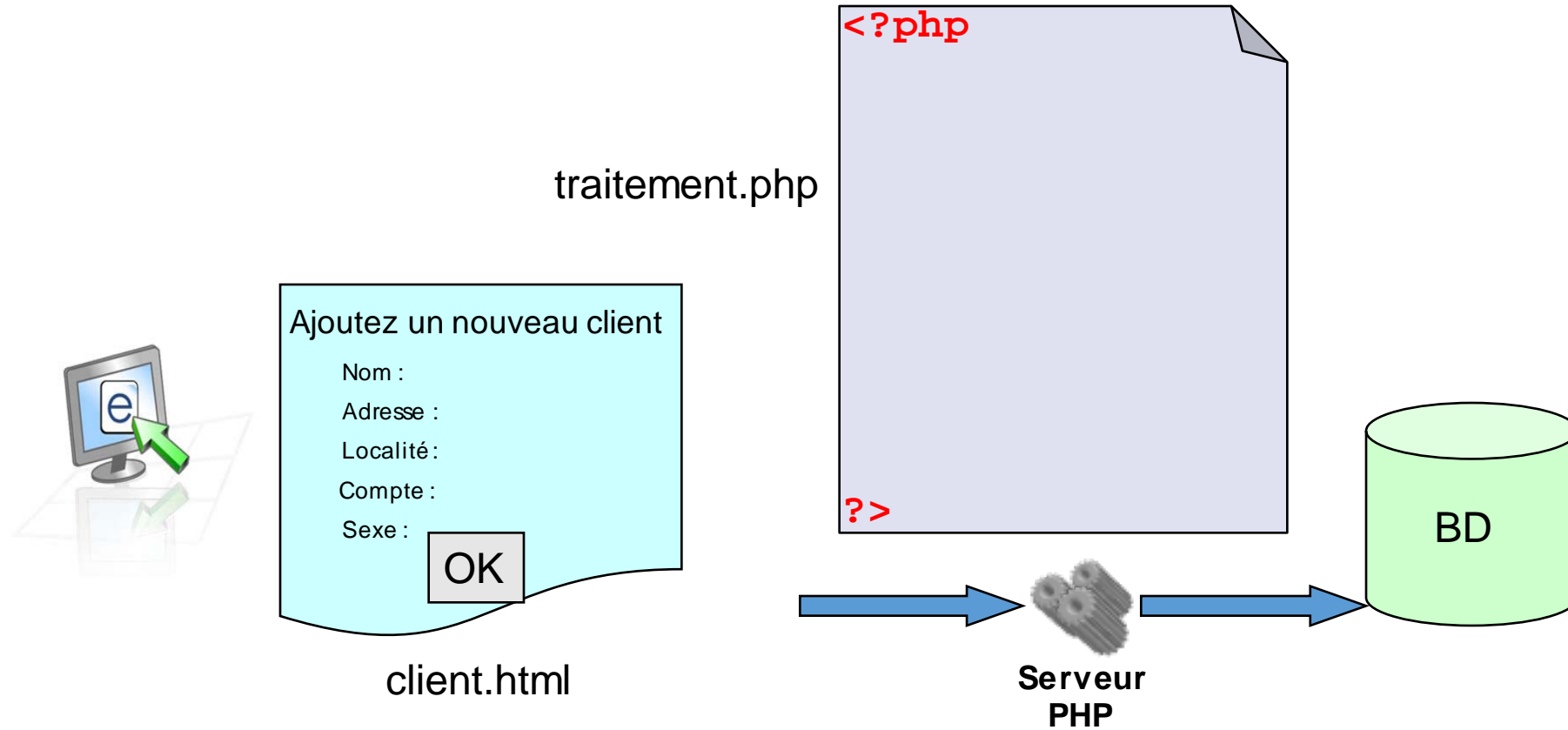
karim.abdelmoula@iit.ens.tn

1^{ère} année Génie Informatique

Chapitre 2 : Traitement des formulaires

Introduction

Le formulaire permet la saisie de données dans une page Web



Réception des données en PHP

- Les formulaires sont des objets d'interface graphiques dotés d'une gestion des données reconnue par tous les navigateurs permettant l'interactivité avec l'internaute.
- **L'un des intérêts majeurs de PHP est sa capacité à exploiter les données** qui sont fournies par un internaute via les **formulaires** HTML.
- PHP **récupère ensuite ces données** pour les insérer dans **une base de données**, pour les exploiter, les afficher, etc.



Retour sur la balise <form>

- Un formulaire est une balise HTML contenant des éléments permettant à un utilisateur de saisir des informations

- Exemple :

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

<input type="text" name="nom" value="Donner votre nom" />

</form>

- **Propriétés de la balise <form>**

- **action** : Sa valeur détermine **l'adresse du script qui recevra le contenu du formulaire**. Cet attribut est requis
 - **enctype** : Sa valeur détermine le type de données envoyées par le formulaire : sont-ce seulement des informations textuelles ou bien y a-t-il également des fichiers joints ? Cet attribut est optionnel et vaut "**application/x-www-form-urlencoded**" par défaut.
 - Si l'on souhaite envoyer des fichiers en plus de texte, il faut donner la valeur "**multipart/form-data**".



Retour sur la balise <form> - suite

- **Id** : Sa valeur est utilisée pour les manipulations du DOM (Document Object Model)
- **method** : Les valeurs peuvent être "**get**" ou bien "**post**"
Cet attribut est optionnel et vaut "get" s'il est omis
- Les deux méthodes de soumission de formulaires :
 - **GET** : Les variables sont transmises par l'URL, ce qui les rend visibles et modifiables très simplement par l'internaute
 - **POST** : Les variables sont transmises de manière cachée : c'est généralement la méthode préférée des développeurs



Les éléments d'un formulaire

- les contrôles **<input>**, **<textarea>**, **<select>**
 - Les **contrôles** sont les éléments HTML qui permettent de saisir des informations.
- La forme que prend la balise **<input>** est définie par la propriété **'type'** :
 - **text** : Permet d'obtenir une petite boîte de saisie de texte dans la page HTML
 - **radio** : Permet d'obtenir une liste dont un seul choix est possible ; il faut utiliser plusieurs boutons *radio* du même nom (propriété *name*)
 - **submit** : Envoie le formulaire
 - **reset** : Rétablit le formulaire à son état d'origine



Les éléments d'un formulaire - suite

- **password** : Même principe que *text*, sauf que le texte n'est pas affiché clairement lors de la saisie
- **image** : Même principe que *submit* mais c'est une image au lieu d'un bouton
- **hidden** : Même principe que *text* mais celui-ci n'est pas affiché dans la page
- **file** : Affiche un bouton permettant de sélectionner un fichier
- **checkbox** : Permet d'obtenir une case à cocher
- **button** : Simplement un bouton ayant la même allure que *submit* ou *reset*





traitement des éléments du formulaire

- Les informations envoyées au moyen d'un formulaire sont stockées dans les tableaux superglobaux de l'environnement PHP
 - Les tableaux utilisés sont :
 - **\$_GET** : Les variables de l'URL (méthode GET)
 - **\$_POST** : Les variables envoyées par la méthode POST
 - **\$_FILES** : Les fichiers envoyés par formulaire
 - **\$_REQUEST** : Les variables \$_POST, \$_GET et \$_COOKIE confondues en une seule ; \$_FILES n'est plus incluse depuis la version 4.3 de PHP

Réception des données en PHP

- Afin de parcourir le contenu d'un formulaire, il est nécessaire de manipuler le tableau **\$_POST** (ou **\$_GET**) comme un tableau associatif :
 - chaque nom de champ (propriété *name* de chaque contrôle) du formulaire est **utilisé comme index de \$_POST** (ou \$_GET)



Cas de la méthode POST

Exemple :

page1.html :

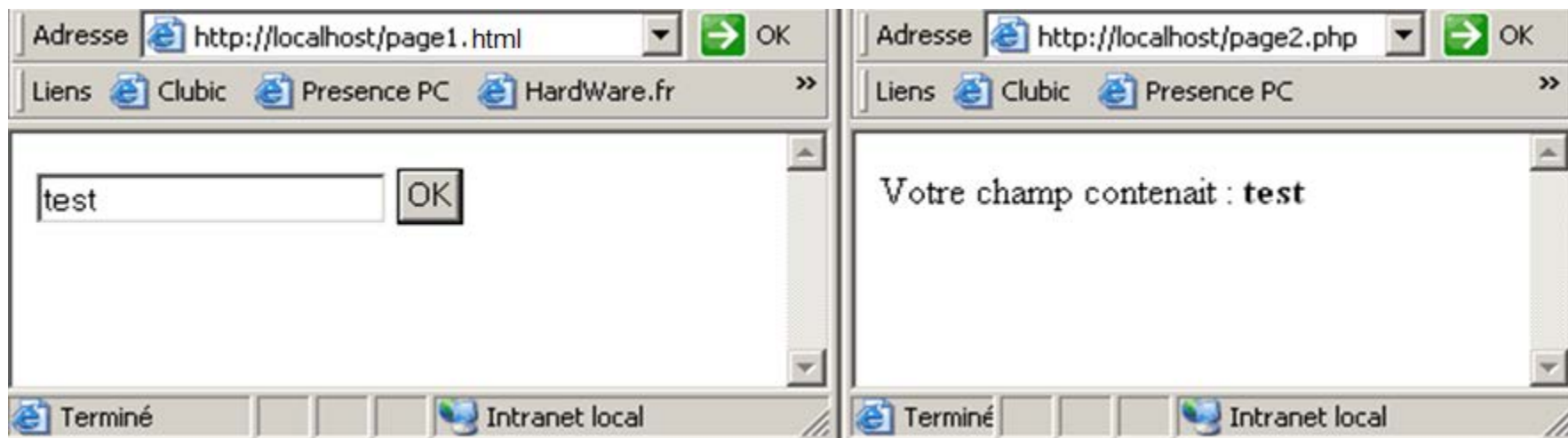
```
<form action="page2.php" method="post">  
<input type="text" name="variable" value="test" />  
<input type="submit" value="OK" />  
</form>
```

page2.php :

```
<?php  
echo "<h2> Votre champ contenait : ". $_POST["variable"]. "</h2>";  
?>
```

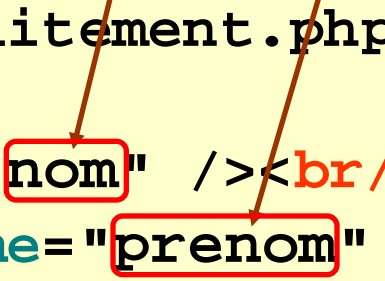


Illustration



Exemple 1

```
<form name="saisie" action="traitement.php"
  method="post">
Nom : <input type="text" name="nom" /><br />
Prénom : <input type="text" name="prenom" /><br />
<input type="submit" value="Valider" />
<input type="reset" value="Effacer" />
</form>
```



Le code de la page cible (traitement.php) :

```
<?php
  echo "Le nom de la  personne".$_POST["nom"];
  echo "Le prénom de la  personne".$_POST["prenom"];
?>
```

Cas de la méthode GET

Exemple :

page1.html :

```
<form action="page2.php" method="get">  
<input type="text" name="variable" />  
<input type="submit" value="OK" />  
</form>
```

page2.php :

```
<?php  
echo "<h2> Votre champ contenait : ". $_GET["variable"]. "</h2>";  
?>
```

Exercice

- Écrire un formulaire multi.html dans lequel l'utilisateur peut saisir un nombre qui représente la valeur du multiplication avant d'appeler le script multi.php.

← → ↻

localhost/Test/multi.html

Entrer un nombre :

Table de Multiplication de 4

- 4 * 0 = 0
- 4 * 1 = 4
- 4 * 2 = 8
- 4 * 3 = 12
- 4 * 4 = 16
- 4 * 5 = 20
- 4 * 6 = 24
- 4 * 7 = 28
- 4 * 8 = 32
- 4 * 9 = 36
- 4 * 10 = 40

[retour vers le formulaire](#)

Solution multi.html

```
<!DOCTYPE html>
<html lang="en">
<head> <title>Formulaire</title></head>
<body>

    <form action="multi.php" method="GET">
        <label for="nbre">Entrer un nombre :</label>
        <input type="number" name="nbre" id="nbre">
        <input type="submit" value="OK">
    </form>

</body>
</html>
```





Débogage avec PHP

- Le débogage est un processus essentiel lors du développement avec PHP. Voici quelques techniques couramment utilisées pour le débogage en PHP :
- **Utilisation de echo**
- **Utilisation de var_dump()**

Utilisation de var_dump() et print_r(): Ces fonctions vous permettent d'afficher les valeurs des variables, des tableaux et des objets à des fins de débogage.

Débogage avec PHP

Voici un exemple d'utilisation de `var_dump()` :

php

```
$name = "John";  
$age = 25;  
$grades = [90, 85, 95];
```

```
var_dump($name);  
var_dump($age);  
var_dump($grades);
```

Résultat

```
string(4) "John"  
int(25)  
array(3) {  
    [0]=>  
        int(90)  
    [1]=>  
        int(85)  
    [2]=>  
        int(95)  
}
```

Requête avec la méthode GET

- Est-ce qu'on peut passer des informations au serveur sans l'utilisation d'un bouton **submit** et la balise **form** ?
- Une requête **GET** peut être initiée par :
 - un formulaire (dont l'attribut method serait configuré avec la valeur GET) :
`<form method="get" action="script.php">`
 - Ou à partir d'un simple lien hypertexte auquel les valeurs à envoyer seront associées à l'URL sous forme de couple variable/valeur après un point d'interrogation ?
 - Si plusieurs couples de variable/valeur doivent être envoyés, ils sont alors séparés par une esperluette : **&**

Exemple :

Page2.php?id=12&nom="mohamed"





Cas de la méthode GET utilisant un lien hypertexte

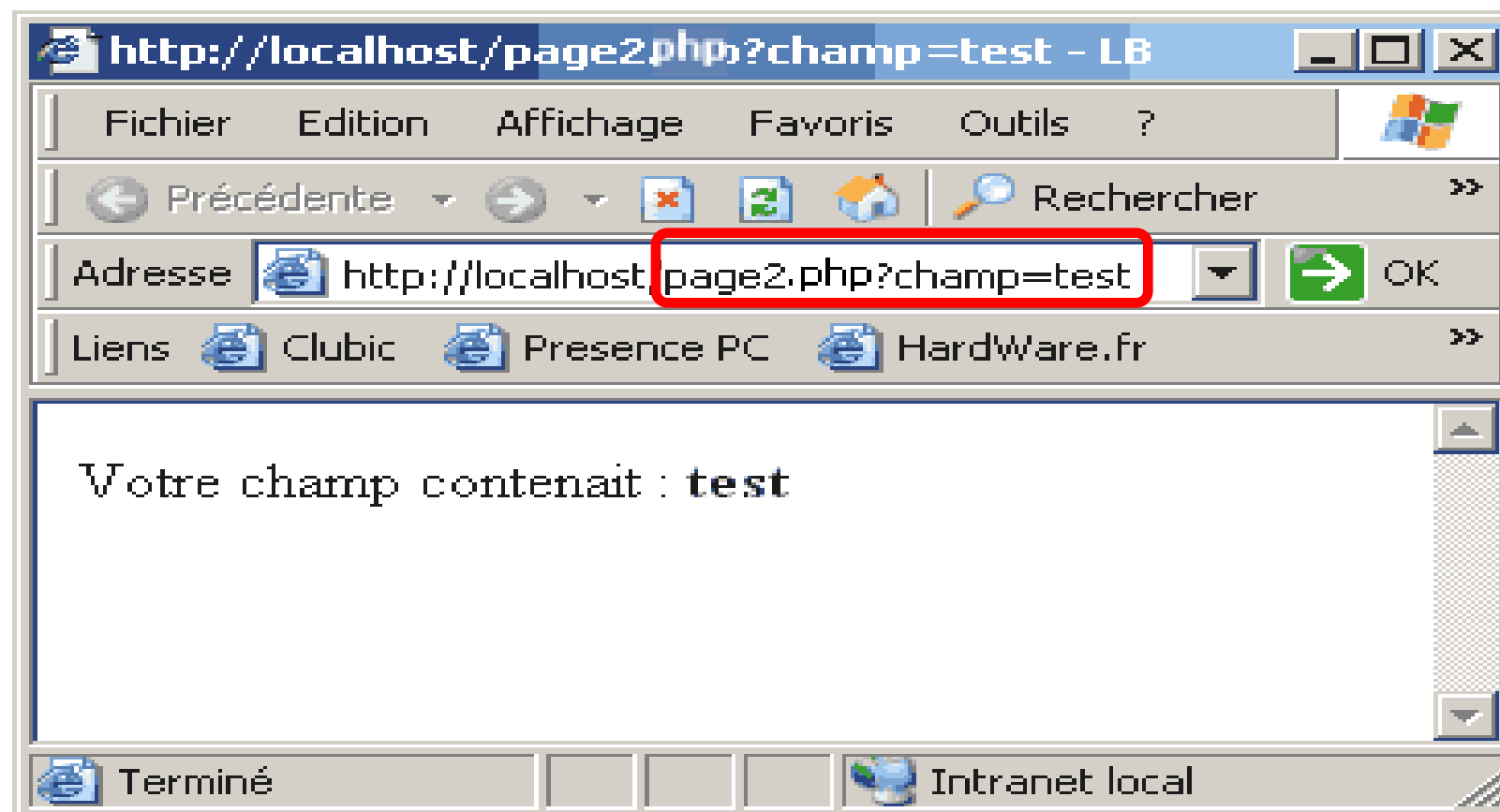
Exemple :

- *Dans le premier fichier (.html):*

```
<a href="page2.php?champ='test'">Cliquez ici </a>
```

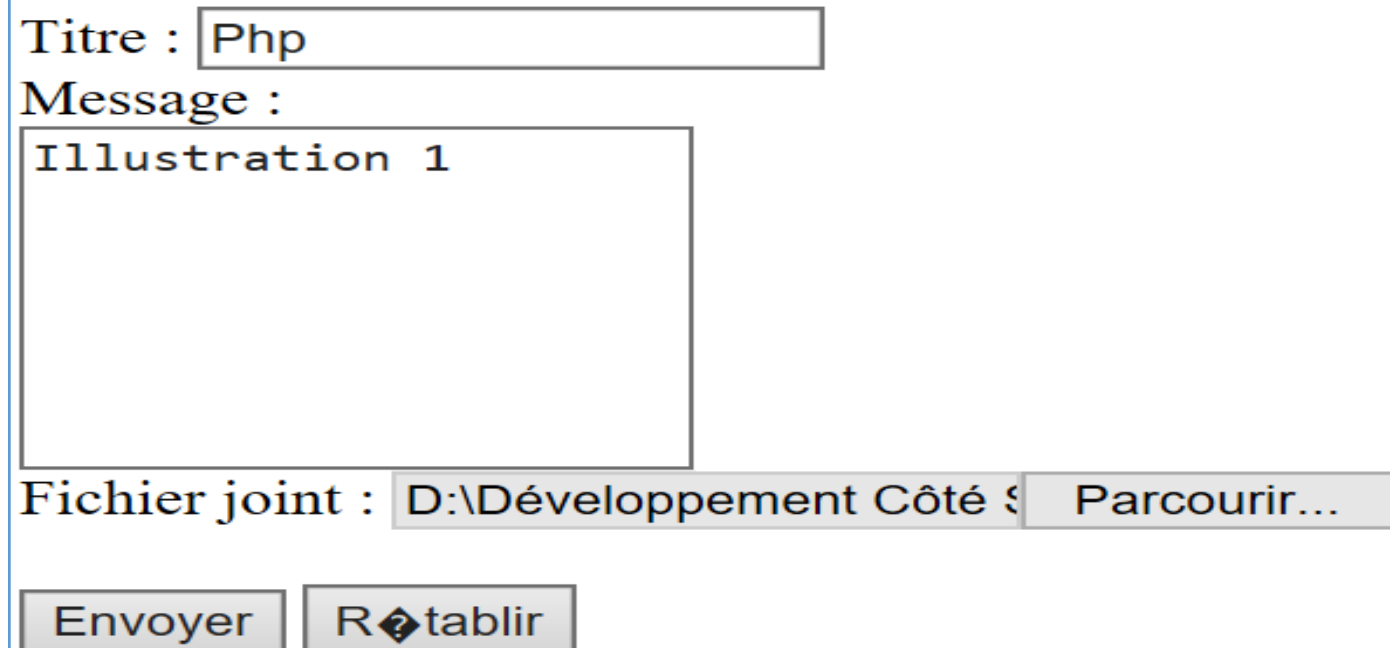
- *Dans le fichier page2.php :*

```
<?php  
echo "<h2> Votre champ contenait : ". $_GET["champ"]."</h2>";  
?>
```



Exemple 2

```
<form method="post" action="traitement.php" enctype="multipart/form-data">  
  <label>Titre : <input type="text" name="title" /></label><br />  
  <label>Message : <br />  
  <textarea name="message" cols="20" rows="7"></textarea></label><br />  
  <label>Fichier joint : <input type="file" name="file" /></label><br /><br />  
  <input type="submit" value="Envoyer" />  
  <input type="reset" value="Rétablir" />  
</form>
```



The screenshot shows a web form with the following elements:

- A text input field labeled "Titre :" containing the text "Php".
- A text area labeled "Message :" containing the text "Illustration 1".
- A file input field labeled "Fichier joint :" showing a file path "D:\Développement Côté S" and a "Parcourir..." button.
- Two buttons at the bottom: "Envoyer" and "Rétablir".



Script « traitement.php » : Afficher le contenu des tableaux superglobaux

```
<?php
if(!empty($_POST)){
    echo '<b>Variables</b> :<br />';
    echo '<pre>';
    print_r($_POST);
    echo '</pre>';
}

if(!empty($_FILES)){
    echo '<b>Fichiers</b> :<br />';
    echo '<pre>';
    print_r($_FILES);
    echo '</pre>';
} ?>
```

Variables :

```
Array
(
    [title] => Php
    [message] => Illustration 1
)
```

Fichiers :

```
Array
(
    [file] => Array
        (
            [name] => TP n°1_IntroPHP.docx
            [type] => application/vnd.openxmlformats-officedocument.wordprocessingml.document
            [tmp_name] => C:\xampp\tmp\phpFE70.tmp
            [error] => 0
            [size] => 109294
        )
)
```



Example 3

```
<form method="post" action="form.php">
  <label>Nom d'utilisateur : <input type="text" name="login" /></label><br />
  <label>Mot de passe : <input type="password" name="password" /></label><br />
<br />
  <input type="submit" value="Envoyer" />
  <input type="reset" value="Rétablir" />
</form>
```

Nom d'utilisateur :

Mot de passe :

Envoyer

Php

Rétablir

js

Array

```
(
    [login] => Ali Salah
    [password] => 123456
)
```

login : Ali Salah

password : 123456





Illustration Exemple3

```
<?php
  if(!empty($_POST)){
    echo '<pre>';
    print_r($_POST); //Afficher le contenu du tableau
    echo '</pre><br />';
    // Récupération normale des informations
    foreach($_POST as $field => $value){
      echo '<b>'.$field.'</b> : '.$value.'<br />';
    }
    echo '<br /><br />';
  } ?>
```

$$\text{\$imc} = \text{\$poids} / (\text{\$taille} * \text{\$taille})$$

Exemple 4

```
<head>
<title>Votre IMC</title>
</head>
<body>
<h1>Déterminez votre IMC et sachez quelle est votre corpulence d'un point de
    vue médical</h1>
<h2>Entrez les données suivantes </h2>
<form name="formulaire" method="post" action="imc.php">
    Entrez votre prénom : <input type="text" name="prenom" /> <br/> Entrez
    votre taille (sous la forme 1.70) :
    <input type="text" name="taille" /> <br/>
    Entrez votre poids (en kilos) : <input type="text" name="poids" /> <br/>
    <input type="submit" name="valider" value="OK" />
</form></body>
```



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.
- Ça 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 possible de donner le même nom à des éléments de saisie de texte s'itérant dans un tableau.



Les valeurs multiples - suite

- Il s'agit d'**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.

Exemple :

```
<form action="script.php" method="POST">
```

```
Bleu:<input type="checkbox" name="choice[ ]" value="bleu" />
```

```
Blanc:<input type="checkbox" name="choice[ ]" value="blanc" />
```

```
...
```

```
</form>
```



Application

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

Quels sont vos couleurs préférées?


```
<input type="checkbox" name="choice[]" value="Red">Rouge
```

```
<table bgcolor="red" width="50"><tr><td>&nbsp;</td></tr></table>
```

```
<input type="checkbox" name="choice[]" value="Blue">Bleu <table  
  bgcolor="blue" width="50"><tr><td>&nbsp;</td></tr></table>
```

```
<input type="checkbox" name="choice[]" value="Green">Vert <table  
  bgcolor="green" width="50"><tr><td>&nbsp;</td></tr></table>
```

```
<input type="checkbox" name="choice[]" value="Black">Noir <table  
  bgcolor="black" width="50"><tr><td>&nbsp;</td></tr></table>
```

```
<input type="submit" value="Voir le résultat!">
```

```
</form>
```



← → ↻ 🏠 ⓘ localhost/Test/checkcouleurs.html

Quels sont vos couleurs préférées?

☐ Rouge

☒ Bleu

☒ Vert

☐ Noir

Voir le résultat!

← ↻ 🏠 ⓘ localhost/Test/couleurs.php

Vos couleurs préférées sont

Blue - Green -



Require vs Include

- Comme en JavaScript, il est possible d'écrire du code PHP dans des fichiers séparés puis de les incorporer dans du code PHP
- Cette possibilité permet notamment de créer une bibliothèque de fonctions d'utilisation courante.
- On donne généralement aux fichiers de code PHP l'extension ".inc.php" (.inc : inclusion)

fonctions.php X

```
fonctions.php
1  <?php
2  function afficher($m)
3  {
4      echo "bonjour $m";
5  }
6  }
7  ?>
```

include.php X

```
include.php
1  <?php
2  include "fonctions.php";
3  afficher("IIT");
4  ?>
```



Require vs Include

- En cas **d'erreur**, si le fichier n'est pas trouvé, `include()` ne génère qu'une alerte, et le script continue. Par contre, `require()` provoque une **erreur** fatale et met fin au script.



Exercice

- Créer un fichier PHP contenant un tableau des alphabets minuscules français et les afficher dans un tableau HTML comme suit :
- **Colonne 1** : contient numéro de l'alphabet en commençant par 1,
- **Colonne 2** : contient l'alphabet,
- **Colonne 3** : contient le type de l'alphabet "consonne" ou "voyelle",
- **Colonne 4** : contient le code ASCII de l'alphabet.
- La ligne d'entête est colorée avec une **couleur d'arrière plan** de votre choix
- Les lignes contenant des voyelles sont colorées avec une **couleur d'arrière plan** différente
- Les lignes contenant des consonnes sont colorées avec une **couleur d'arrière plan** différente

numéro	alphabet	type	ascii
1	a	voyelle	97
2	b	consonne	98
3	c	consonne	99
4	d	consonne	100
5	e	voyelle	101
6	f	consonne	102
7	g	consonne	103
8	h	consonne	104
9	i	voyelle	105
10	j	consonne	106
11	k	consonne	107
12	l	consonne	108



Style.css

```
table{
    border-collapse: collapse;
    margin: auto;
}
th{
    background-color: rgb(0, 255, 157);
}
td,th{
    border-top: 1px solid black;
    border-bottom: 1px solid black;
}
.voy{
    background-color: rgb(0, 132, 255);
}
.cons{
    background-color: #95eb73;
}
```