
2018 / 2019

# Partie 1


# Présentation

Laurent Dargent

Poste : Lead Developer

Entreprise : Kaliop (Montpellier)


# PARTIE 1

## SOMMAIRE

01

A. Origine
B. Historique
C. Atouts

# Introduction

## D’où provient PHP ?

- Langage de **script** exécuté côté **serveur**
- Acronyme récursif : **P**HP: **H**ypertext **P**reprocessor (origine: **P**ersonal **H**ome **P**age)
- Créé et utilisé en **1994** par **Rasmus Lerdorf** pour ses besoins personnels
- A l’origine une boîte à outils, puis un vrai langage de programmation

## Historique de PHP

- **1995 :** Rasmus Lerdorf publie PHP/FI
- **1997 :** Réécriture en version 3 par Zeev Suraski et Andi Gutmans (fondateurs de Zend)
- **2000 :** Publication de la version 4 (langage principalement procédural)
- **2002 :** Plus de 8 millions de sites web l'utilise
- **2004 :** Publication de la version 5 (intégration d'un réel concept de POO)
- **2010 :** Abandon de la version 6 au profit d'une version 5.4
- **2013 :** Plus de 244 millions de sites web l'utilise
- **2015 :** Sortie du très attendu PHP 7  (deux fois plus rapide)
- **Aujourd'hui :**
  - Versions supportées : PHP 5.6 et PHP 7
  - Plus de 82% des sites sont écrits en PHP

## Atouts du langage

- Rapide à apprendre et à utiliser (*en apparence*)
- Pas de compilation des sources (interprété)
- Multi-plateforme
- Entreprise éditrice (Zend)
- Communauté très forte (bugfix, extensions, librairies, frameworks,...)

# Partie 1 - Introduction

## Atouts

02

A. Le protocole HTTP - Définition
B. Le protocole HTTP - Fonctionnement
C. Composants d'un serveur PHP

# Concepts web

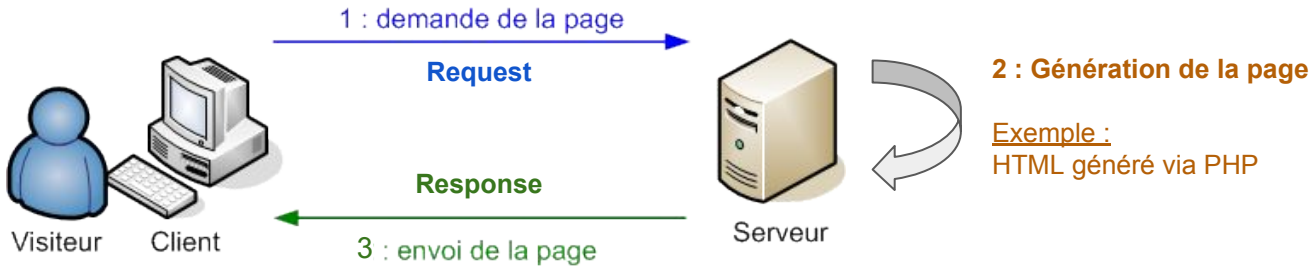
## **H**yper**T**ext **T**ransfer **P**rotocol

- Protocole de **communication** machine à machine (ex : navigateur + serveur)
- Echange de **documents** (réception + envoi)
- Utilisé massivement sur le web (sites internet, objets connectés, ...)

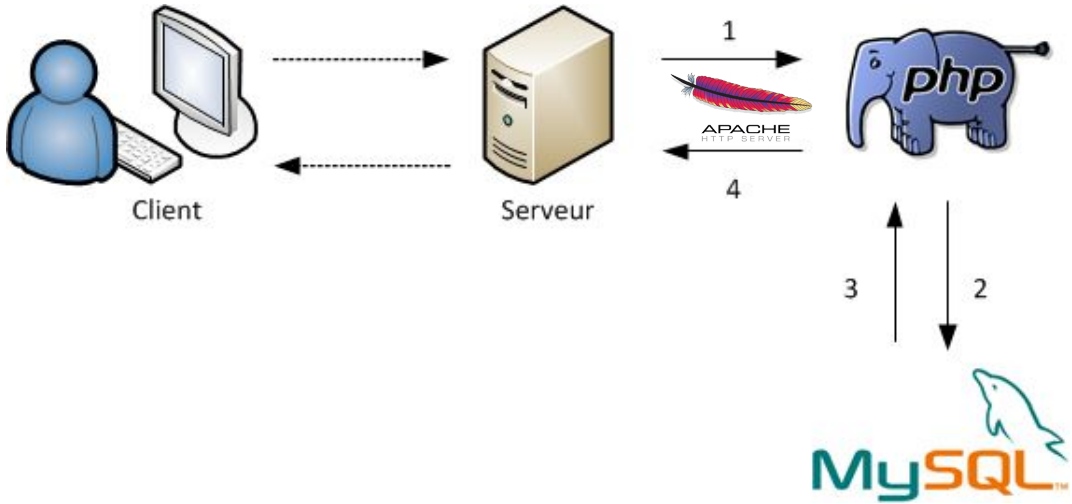
## **H**yper**T**ext **T**ransfer **P**rotocol

- Basé sur le principe de **Request** et **Response**
- Utilisation de méthodes de communication
  - **GET** : Demande d'un document
  - **POST** : Envoi de données vers le serveur
  - …
- Constitué d'en-têtes "**headers**" et de données "**body**"

## **H**yper**T**ext **T**ransfer **P**rotocol



## Infrastructure PHP



# 03

A. Syntaxe
B. Variable, constante et typage
C. Les tableaux

# Les bases

# Partie 1 - Les bases

Let’s go !

## Balises script

Ouverture et fermeture d'un script

```php
<?php
...
?>
```

## Commentaires

Écriture de commentaire

```php
<?php
// Mon premier commentaire

/* Mon second commentaire
sur plusieurs lignes.
Oui c'est possible ! */
?>
```

# Partie 1 - Les bases

Syntaxe

## Hello world !

Affichage de texte

```php
<?php
echo "Salut à toi dev PHP !";
print("Comment vas-tu ?");
?>
```

Toute instruction PHP se termine par un “;”

## Inclusion dans d’autres scripts

Insertion au sein de code HTML

```php
<html>
    <head>...</head>
    <body>
        <ul>
            <li><?php echo "Pomme"; ?></li>
            <li><?php echo "Poire"; ?></li>
        </ul>
    </body>
</html>
```

# Variables

Déclaration d'une variable

```php
<?php
$prenom = "Eric";
$age = "28";

echo 'Je m\'appelle '.$prenom.'<br />';
echo "<div>J'ai $age ans</div>";
?>
```

- Une variable commence toujours par un "**$**"
- Stockée en mémoire vive
- Durée de vie limitée au script en cours
- Pas de limite de nombre de variable

## Constantes

Déclaration d'une constante

```php
<?php
define('SERVER_NAME', 'vm-front-php.home.fr');
echo SERVER_NAME;
?>
```

- Pas de préfixe pour le nom
- Valeur non modifiable
- Nommée en majuscule par convention
- Type simple uniquement (string, int, array, ...)

## Constantes

Déclaration d'une constante de tableau

```php
<?php
define('SERVER_IDS', array(1, 2, 3));
var_dump(SERVER_IDS);
?>
```

- Introduit en PHP 7

## Le typage

- Langage à typage **dynamique**
  - Pas nécessaire de définir le type lors de la déclaration d'une variable
  - Le type est définie à la volée (lors de l'affectation de la valeur)
- Possibilité de forcer le type (cast ou transtypage)
- Possibilité de tester le type d'une variable

# Partie 1 - Les bases

Variable, constante et typage

## Le typage

Déclaration, transtypage et contrôle d'une variable

```php
<?php
$var = "12";                // Déclaration d'une chaîne "12"
var_dump($var);             // Affichage de la variable $var
$var = (int) $var;          // Transtypage de la variable en 'int'
var_dump($var);             // Affichage de la variable $var
var_dump(is_string($var)); // Test si $var est une chaîne
var_dump(is_int($var));    // Test si $var est un entier
gettype($var);              // Affiche le type de la variable $var
?>
```

Résultat

```
string(2) "12"

int(12)
bool(false)
bool(true)
integer
```

## Les types de données

- **string :** Chaîne de caractères (ex : “Je suis du texte”)
- **int :** Nombre entier (ex : 1, 2, 12, 58, ...)
- **float :** Nombre décimaux (ex : 12,58)
- **boolean :** Booléen (true/false)
- **array :** Tableau de données
- **object :** Objet (Programmation orientée objet)
- **resource :** Ressource externe (ex : connexion à une base de données)
- ***null :*** Pour des valeurs de type non définies

## Des tableaux simples et complexes à la fois

- Déclaration très simple
- Multi-dimensionnement facile à mettre en oeuvre
- Nombreuses fonctions de manipulation natives
- Clé numérique et associative

# Partie 1 - Les bases

Les tableaux

## Déclaration

Tableau avec clés numériques

```php
<?php
$tab = array(); // Déclaration d'un tableau vide
var_dump($tab);

// Déclaration d'un tableau avec contenu
$tab = array('guitare', 'batterie', 'clavier');

// Ajout d'un élément au tableau
$tab[] = 'basse';
var_dump($tab);
?>
```

Résultat

```
array(0) {
}
array(4) {
  [0]=>
  string(7) "guitare"
  [1]=>
  string(8) "batterie"
  [2]=>
  string(7) "clavier"
  [3]=>
  string(5) "basse"
}
```

# Partie 1 - Les bases

Les tableaux

## Déclaration

Tableau avec clés associatives

```php
<?php
$tab = array();

// Déclaration d'un tableau avec contenu
$tab['morceau'] = "Feel good inc.";
$tab['musicien'] = array('Albarn', 'Hewlett');
$tab['nb_albums'] = 4;

// Ajout d'un élément au tableau
$tab['musicien'][] = 'Niccals';
var_dump($tab);
?>
```

Résultat

```
array(3) {
  ["morceau"]=>
  string(14) "Feel good inc."
  ["musicien"]=>
  array(3) {
    [0]=>
    string(6) "Albarn"
    [1]=>
    string(7) "Hewlett"
    [2]=>
    string(7) "Niccals"
  }
  ["nb_albums"]=>
  int(4)
}
```

## Fonctions de tableau

Exemple de fonctions natives

```php
<?php
$tab = array('Albarn', 'Hewlett', 'Niccals');
var_dump(in_array('Hewlett', $tab));
var_dump(in_array('Carlos', $tab));

$tab2 = array_merge($tab, array('Hobbs', 'Snoop Dog'));
var_dump($tab2);

array_diff(...) // Calcul la différence entre deux tableaux
array_pop(...) // Dépile un élément du tableau
...
?>
```

Résultat

```
bool(true)
bool(false)

array(5) {
  [0]=>
  string(6) "Albarn"
  [1]=>
  string(7) "Hewlett"
  [2]=>
  string(7) "Niccals"
  [3]=>
  string(5) "Hobbs"
  [4]=>
  string(9) "Snoop Dog"
}
```

# Partie 1 - Les bases

## Les tableaux

04

A. Les conditions
B. Les boucles
C. Les arrêts

# Les structures de contrôles

# Partie 1 - Les structures de contrôles

## Les conditions

### If, then, else

```php
<?php
if (expression) {
    // Bloc si expression vraie
}

if (expression) {
    // Bloc si expression vraie
} else {
    // Bloc si expression fausse
}
?>
```

Exemple

```php
<?php
if ($a == $b) {
    echo $a;
} else {
    echo $b;
}
?>
```

# Partie 1 - Les structures de contrôles

## Les conditions

### If, then, else

```php
<?php
if (expression) {
    // Bloc si expression vraie
} elseif (expression) {
    // Bloc si expression vraie
} else {
    // Bloc si expression fausse
}
?>
```

# Partie 1 - Les structures de contrôles

Les conditions

## Switch, case

```php
<?php
switch (expression) {
    case valeur1:
        ...
    case valeur2:
        ...
    default :
        ...
}
?>
```

Exemple

```php
<?php
switch ($planete) {
    case "venus":
        echo "2nd planète";
        break;
    case "mars":
        echo "4ème planète";
        break;
    default :
        echo "Planète inconnue";
        break;
}
?>
```

# Partie 1 - Les structures de contrôles

## Les conditions

### Opérateur ternaire

```php
<?php
$var2 = (expression) ? // vrai : // faux;
?>
```

Exemple

```php
<?php
$var2 = (empty($var)) ? 'défaut' : $var;
?>
```

# Partie 1 - Les structures de contrôles

## Les boucles

### La boucle “for”

```php
<?php
for (initialisation; expression; incrémentation) {
    ...
}
?>
```

Exemple

```php
<?php
// boucle simple
for ($i = 1; $i <= 10; $i++) {
    echo $i;
}

// boucle sur tableau
$beers = array('blonde', 'brune', 'rousse');
for ($i = 0; $i < count($beers); $i++) {
    echo $beers[$i];
}
?>
```

# Partie 1 - Les structures de contrôles

Les boucles

## La boucle “while”, “do..while”

```php
<?php
// Exécuté tant que l'expression est 'vraie'
while (expression) {
    ...
}

// Toujours exécuté une première fois
do {
    ...
} while (expression);

/* L'expression doit changer d'état
afin d'éviter les boucles infinies ! */
?>
```

Exemple

```php
<?php
// Affiche les valeurs de 1 à 10
$i = 1;
while ($i <= 10) {
    echo $i++;
}

// Affiche '0'
$i = 0;
do {
    echo $i;
} while ($i > 0);
?>
```

# Partie 1 - Les structures de contrôles

Les boucles

## La boucle “foreach” pour les tableaux

```php
<?php
// Boucle sur les valeurs du tableau
foreach (array_expression as $value){
    ...
}

// Idem + récupération de l’index de la valeur
foreach (array_expression as $key => $value){
    ...
}
?>
```

Exemple

```php
<?php
$beers = array('blonde', 'brune', 'rousse');
foreach ($beers as $beer) {
    echo $beer;
}

$translations = array(
    'door' => 'porte',
    'window' => 'fenêtre',
    'desktop' => 'bureau'
);
foreach ($translations as $eng => $fre) {
    echo $eng."->".$fre;
}
?>
```

## Les arrêts “break” et “exit”

- **break** : Permet la sortie d’un bloc
- **exit** : Permet l’arrêt du programme

Exemple

```php
<?php
$i = 0;
while (++$i) {
    switch ($i) {
    case 5:
        echo "5";
        break 1;  // Termine uniquement le switch.
    case 10:
        echo "10";
        break 2;  // Termine le switch et la boucle while.
    default:
        break;
    }
}
?>
```

05

A. Arithmétique
B. Logique
C. Comparaison
D. Affectation

# Les opérateurs

# Partie 1 - Les opérateurs

## Arithmétique

| Exemple |
| --- |
| -$a |
| $a + $b |
| $a - $b |
| $a * $b |
| $a / $b |
| $a % $b |
| $a ** $b |

# Partie 1 - Les opérateurs

## Logique

| Opérateur | Exemple |
|---|---|
| && / and | $a && $b (l'un et l'autre) |
| \|\| / or | $a \|\| $b (l'un, l'autre ou les deux) |
| ! | ! true (donne false) |
| xor | $a xor $b (l'un ou l'autre mais pas les deux) |

# Partie 1 - Les opérateurs

## Comparaison

| Opérateur | Description |
| --- | --- |
| == | Egalité de deux valeurs |
| === | |
| != | Différence de deux valeurs |
| !== | |
| <, >, <=, >= | Infériorité, supériorité, ... |

# Partie 1 - Les opérateurs

## Comparaison

| Opérateur | Description |
|---|---|
| == | Egalité de deux valeurs |
| === | Egalité de deux valeurs et leur type |
| != | Différence de deux valeurs |
| !== | Différence de deux valeurs et leur type |
| <, >, <=, >= | Infériorité, supériorité, ... |

## Nouveautés PHP 7

| Opérateur | Nom | Description |
|---|---|---|
| <=> | Combiné | |
| ?? | Union nul | |

# Partie 1 - Les opérateurs

Comparaison

## Nouveautés PHP 7

| Opérateur | Nom | Description |
|---|---|---|
| <=> | Combiné | Comparateur d'expression (spaceship)<br>• a == b => 0<br>• a > b => 1<br>• a < b => -1 |
| ?? | Union nul | Renvoi la première occurrence non nulle<br>Ex : $limit = $query['limit'] ?? 10; |

# Partie 1 - Les opérateurs

## Affectation

| Opérateur | Description | Exemple |
|---|---|---|
| + | Addition | $a += 5; |
| . | Concaténation | $a .= “test” |
| ++ | Incrémentation | $a++ |
| -- | Décrémentation | $a-- |

06

A. Déclaration de fonction
B. Typage
C. Les fonctions natives

# Les fonctions

# Partie 1 - Les fonctions

## Déclaration de fonction

- Permet de rendre du code réutilisable
- Déclaration simple
- Pas de typage nécessaire pour les arguments
- Pas de typage nécessaire pour le retour de la fonction
- Arguments facultatifs possible
- Passage d’arguments par référence

# Partie 1 - Les fonctions

## Déclaration de fonction

```php
<?php
// Déclaration simple
function maFonction($arg1, $arg2, ...) {
    // Corps de la fonction
}

// Déclaration avec valeur de retour
function maFonction($arg1, $arg2, ...) {
    // Corps de la fonction
    return $value;
}
?>
```

```php
<?php
// Déclaration simple
function display($content) {
    echo $content;
}
display("Hola !"); // Affiche 'Hola !'

// Déclaration avec valeur de retour
function display($content) {
    return $content;
}
echo display("Hola !"); // Affiche 'Hola !'
?>
```

# Partie 1 - Les fonctions

## Déclaration de fonction

```php
<?php
// Argument facultatif
function maFonction($arg1, $arg2 = null) {
    // Corps de la fonction
}

// Passage d'argument par référence
function maFonction($arg1, &$arg2) {
    // Corps de la fonction
}
?>
```

```php
<?php
function meteo($value, $type = "°C") {
    return "Il fait ".$value.$type;
}
echo meteo(35);       // Affiche 'Il fait 35°C'
echo meteo(95, "°F"); // Affiche 'Il fait 95°F'

function addHtmlMarkup(&$string, $markup) {
    $string = '<'.$markup.'>'.$string.'</'.$markup.'>';
}

$html = 'Hello !';
addHtmlMarkup($html, 'div');
echo $html; // Affiche '<div>Hello !</div>';
?>
```

# Partie 1 - Les fonctions

## Typage

### Les arguments

- Typage de classe (PHP 5)
- Typage scalaire (PHP 7)

Exemple

```php
<?php
// Argument typé par classe
function typageClass(MyClass $object) {
    ...
}

// Argument par typage scalaire (string, int, float, bool)
function typageScalaire(int $entier) {
    ...
}
?>
```

# Partie 1 - Les fonctions

## Typage

### Valeur de retour

- Introduit en PHP 7
- Types identiques aux arguments
- Fonction sans retour

Exemple

```php
<?php

// Fonction avec retour typé
function typageScalaire(int $entier): string {
    return "Number : " . $entier;
}

// Fonction avec retour typé
function sansRetour(int $entier): void {
    ... // Pas de return
}


?>
```

# Partie 1 - Les fonctions

## Les fonctions natives

- Manipulation de chaînes
- Manipulation de tableau
- Recherche d'éléments
- ...

Exemple

```php
<?php
echo strlen("lachainedecaracteres"); // 20
echo str_replace("a","o","Lalala"); // 'Lololo'
echo trim("  Salut le monde   "); // 'Salut le monde'
echo strpos("abcdef","e"); // 4
echo in_array('toto', array('titi', 'toto', 'tata')); // true
...
?>
```

# Partie 1 - Les fonctions

## Les fonctions natives

07

A. Définition
B. Les variables globales
C. Les variables statiques

# Portée des variables

# Partie 1 - Portée des variables

Définition

## Accessibilité des variables

- Une variable est locale au bloc dans lequel elle a été déclarée
- Possible de modifier la portée d’une variable

# Partie 1 - Les bases

Variables globales

## Variable globale

```php
<?php
$var1 = "manteau";
$var2 = "chemise";

function test() {
    global $var2;
    echo $var1;
    echo $var2;
    $var2 = "pantalon";
}

test();
echo $var2;
?>
```

Résultat

```
chemise
pantalon
```

# Partie 1 - Les bases

Variables statiques

## Variable static

```php
<?php
    function incrementation() {
        static $i = 0; // Initialisation de la variable au premier appel de la fonction
        $i++;
        echo $i;
    }

    incrementation(); // Affiche '1'
    incrementation(); // Affiche '2'
    incrementation(); // Affiche '3'
    incrementation(); // Affiche '4'
    incrementation(); // Affiche '5'
?>
```

# 08

A. Avantages
B. Fonctions d'inclusion

# Inclusion de script

# Partie 1 - Inclusion de script

Avantages

## Inclure un script dans son script principale

- Permettre une réutilisation du code
- Permettre de bien séparer son code
- Script principale plus lisible
- Inclusion de librairie tierce

# Partie 1 - Inclusion de script

## Fonctions d’inclusions

| Fonction | Description |
| --- | --- |
| require() | Permet d’inclure un script php. |
| include() | |
| require_once() | Identique à require() mais inclut le fichier une seule fois si ce dernier a déjà été inclu précédemment. |
| include_once() | Identique à include() mais inclut le fichier une seule fois si ce dernier a déjà été inclu précédemment. |

# Partie 1 - Inclusion de script

## Fonctions d'inclusions

| Fonction | Description |
| --- | --- |
| require() | Permet d'inclure un script php. Produit une erreur fatale si une erreur se produit. |
| include() | Identique à require() mais ne provoque uniquement un warning. |
| require_once() | Identique à require() mais inclut le fichier une seule fois si ce dernier a déjà été inclu précédemment. |
| include_once() | Identique à include() mais inclut le fichier une seule fois si ce dernier a déjà été inclu précédemment. |

## Exemple d’inclusions

```php
<?php
// Inclusions de fichiers
include('header.php');
require('menu.php');
include_once('right_column.php');
require_once(footer.php');
?>
```

09

A. Envoi via url
B. Envoi via formulaire
C. Envoi de fichiers
D. Sécurité

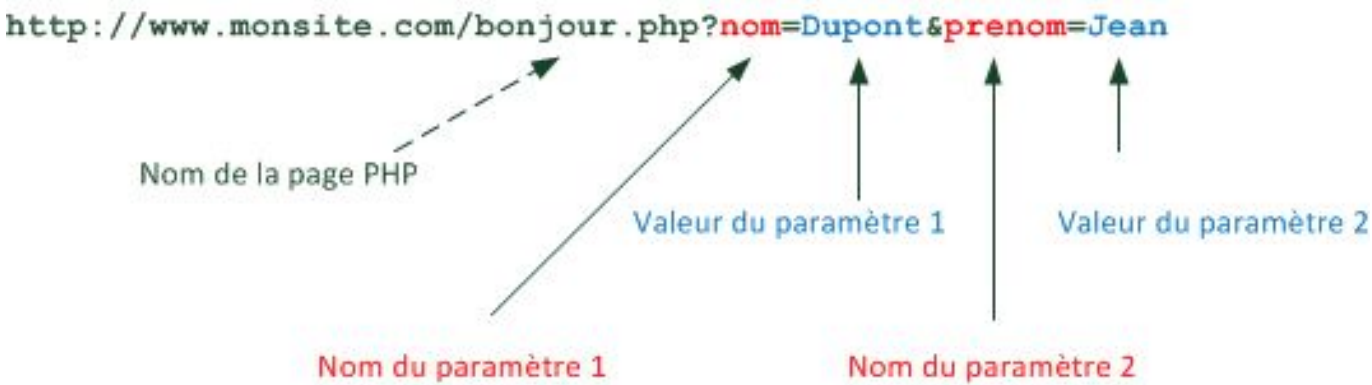
# Envoi de données

## Envoi de données via l'url

- Les données sont transmises dans l'URL du navigateur
- Chaque données est associée à un paramètre
- Utilisation de la méthode GET du protocole HTTP
- A utiliser pour des envois limités de données
- Pas de limites de paramètre mais une taille d'URL limitée
- Les données peuvent être modifiées par l'internaute

# Partie 1 - Envoi de données

## Envoi via url



# Partie 1 - Envoi de données

## Envoi via url

```
http://www.monsite.com/bonjour.php?nom=Dupont&prenom=Jean
```

```php
// Affichage des variables
<p>Bonjour <?php echo $_GET['prenom'] . ' ' . $_GET['nom']; ?> !</p>
// Affiche 'Bonjour Jean Dupont !'

<?php
// Contrôle de la présence des variables
if (isset($_GET['prenom']) and isset($_GET['nom'])) {
    echo 'Bonjour ' . $_GET['prenom'] . ' ' . $_GET['nom'] . ' !';
}
?>
```

# Partie 1 - Envoi de données

## Envoi via url

### Possibilité de réaliser des liens avec paramètres

```
<!-- index.php -->

<a href="bonjour.php?nom=Dupont&amp;prenom=Jean">Dis-moi bonjour !</a>
```

## Envoi de données via un formulaire HTML

- Les données sont transmises dans le corps de la requête HTTP
- Les données sont saisies au sein des éléments HTML (input, checkbox, ...)
- Utilisation de la méthode POST du protocole HTTP
- Taille limite d'envoi de données définie dans la configuration PHP

# Partie 1 - Envoi de données

Envoi via formulaire

## Envoi de données via un formulaire HTML

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Saisie de nom</title>
</head>
<body>
    <form method="post" action="bonjour.php">
        Nom: <input type="text" name="nom" /><br />
        Prénom: <input type="text" name="prenom" />
        <input type="submit" />
    </form>
</body>
</html>
```

# Partie 1 - Envoi de données

Envoi via formulaire

## Envoi de données via un formulaire HTML

```php
// Affichage des variables
<p>Bonjour <?php echo $_POST['prenom'] . ' ' . $_POST['nom']; ?> !</p>
// Affiche 'Bonjour Jean Dupont !'

<?php
// Contrôle de la présence des variables
if (isset($_POST['prenom']) and isset($_POST['nom'])) {
    echo 'Bonjour ' . $_POST['prenom'] . ' ' . $_POST['nom'] . ' !';
}
?>
```

## Envoi de fichiers via un formulaire HTML

- Les données sont saisies au sein d'éléments HTML (type 'file')
- Utilisation de la méthode POST du protocole HTTP
- La méthode d'encodage HTTP du formulaire doit être précisée
- Taille limite d'envoi de fichier définie dans la configuration PHP

# Partie 1 - Envoi de données

Envoi de fichiers

## Envoi de fichiers via un formulaire HTML

```
<!DOCTYPE html>
<html>
<head>...</head>
<body>
<form action="envoi.php" method="post" enctype="multipart/form-data">
    <p>
        Formulaire d'envoi de fichier :<br />
        <input type="file" name="monfichier" /><br />
        <input type="submit" value="Envoyer le fichier" />
    </p>
</form>
</body>
</html>
```

## Envoi de fichiers via un formulaire HTML

- Le fichier est placé temporairement sur le serveur
- Le traitement est plus complexe que pour l'envoi de données
- Utilisation de la variable $_FILES
- La sauvegarde du fichier est réalisé en PHP

## Envoi de fichiers via un formulaire HTML

| Variable | Description |
|---|---|
| $_FILES['monfichier']['name'] | Contient le nom du fichier envoyé par le visiteur. |
| $_FILES['monfichier']['type'] | Indique le type du fichier envoyé. |
| $_FILES['monfichier']['size'] | Indique la taille du fichier envoyé (en octets). |
| $_FILES['monfichier']['tmp_name'] | Emplacement temporaire du fichier sur le serveur |
| $_FILES['monfichier']['error'] | Code d'erreur permettant de connaître l'état de l'envoi.<br>Vaut 0 s'il n'y a pas d'erreur. |

# Partie 1 - Envoi de données

Envoi de fichiers

## Envoi de fichiers via un formulaire HTML

```php
<?php
// Testons si le fichier a bien été envoyé et s'il n'y a pas d'erreur
if (isset($_FILES['monfichier']) && $_FILES['monfichier']['error'] == 0) {

    // On peut valider le fichier et le stocker définitivement
    $fileName = basename($_FILES['monfichier']['name']);
    move_uploaded_file($_FILES['monfichier']['tmp_name'], 'uploads/' . $fileName);
    echo "L'envoi a bien été effectué !";
}
?>
```

## Sécurité des données transmises

- **Ne jamais faire confiance aux données envoyées**
- Risque d'injection XSS (envoi de code HTML, Javascript)
- Risque d'injection PHP
- Risque d'injection SQL
- ...

# Partie 1 - Envoi de données

## Sécurité

# Partie 1 - Envoi de données

Sécurité

## Eviter les failles XSS

```php
<?php
/* Appelle d'une URL avec paramètre
   bonjour.php?nom=<b>Laurent</b> */

// affiche 'Laurent'
echo $_GET['nom'];

// affiche '<b>Laurent</b>'
echo htmlspecialchars($_GET['nom']);
?>
```

# Partie 1 - Envoi de données

Sécurité

## Sécuriser l'envoi de fichier

```php
<?php
// Testons si le fichier a bien été envoyé et s'il n'y a pas d'erreur
if (isset($_FILES['monfichier']) && $_FILES['monfichier']['error'] == 0) {
    // Testons si l'extension est autorisée
    $fileDatas = pathinfo($_FILES['monfichier']['name']);
    $extensionUpload = $fileDatas['extension'];
    $allowed_extensions = array('jpg', 'jpeg', 'gif', 'png');
    if (in_array($extensionUpload, $allowed_extensions)) {
        // On peut valider le fichier et le stocker définitivement
        $fileName = basename($_FILES['monfichier']['name']);
        move_uploaded_file($_FILES['monfichier']['tmp_name'], 'uploads/' . $fileName);
        echo "L'envoi a bien été effectué !";
    }
}
?>
```

10

A. Définition
B. Les variables

# Variables superglobals

# Partie 1 - Variables superglobals

## Définition

# Partie 1 - Variables superglobals

## Définition

- Variables globales à l'ensemble du script
- Automatiquement générées par PHP
- Variables de type 'array'
- Accessible directement via leurs nom
- Contiennent un grand nombre d'informations
  - Informations serveurs
  - Données transmises
  - Informations sur la requête HTTP
  - Identification de l'utilisateur
  - ...

# Partie 1 - Variables superglobals

## Les variables

| Variable | Description |
|---|---|
| $_SERVER | Valeurs envoyées par le serveur (ex : adresse IP, nom du script, …) |
| $_ENV | Variable d'environnement du serveur (peu utilisé) |
| $_GET | Données envoyées via l'URL (ou par la méthode 'get' d'un formulaire) |
| $_POST | Données postées via un formulaire HTML |
| $_COOKIE | Contient les cookies utilisateurs |
| $_REQUEST | Concaténation de $_GET, $_POST et $_COOKIE |
| $_SESSION | Contient les variables de session (utile pour conserver de l'information) |
| $_FILES | Contient les informations sur les fichiers envoyés |

11

A. Cookie
B. Session

# Authentification

# Partie 1 - Authentification

Cookie

## Qu’est ce qu’un cookie ?

- Permet le stockage d’une information utilisateur sur le poste client
- Associé à un site (nom de domaine)
- Possède une durée de vie
- Peut stocker plusieurs informations
- Doit être créé avant l’affiche du premier caractère

```php
<?php
// Déclarer / modifier un cookie
setcookie('login', 'Tom', time() + 365*24*3600);

// Lire un cookie
echo $_COOKIE['pseudo'];
?>
```

IUT MONTPELLIER-SETE

# Partie 1 - Authentification

Session

## Qu’est ce qu’une session ?

- Permet le stockage d’une information utilisateur sur le serveur
- Permet de conserver des variables de page en page
- Pas de limite de taille
- Permet l’authentification utilisateur
- La session est unique par utilisateur

# Partie 1 - Authentification

Session

## Manipuler une session

```php
<?
// Démarrage de la session
session_start();

// Définition d'une variable
$_SESSION['login'] = 'julien';
?>
```

```php
<?
// Démarrage de la session
session_start();

// Lecture d'une variable de session
echo $_SESSION['login'];
?>
```

```php
<?php
session_start();

// Suppression de la session
session_destroy();
?>
```

12

A. Description et problématique
B. Les mauvais exemple
C. Une standardisation

# Ecriture et standard

# Partie 1 - Ecriture et standard

Description et problématique

## Ecrire du code PHP

- L’écriture du code est propre à chacun
  - indentation, espace, ...
  - nommage de variable, de fonctions, d’objets...
- Code de projet non homogène
- Lisibilité et compréhension du code complexifiées
- Risque d’erreurs de code

# Partie 1 - Ecriture et standard

## Les mauvais exemples

```php
<?php
$ma_var="toto";
  $maVar2="tutu";
$tmp     ="15";

if($ma_var==$maVar2 )
    $tmp+=2;
    $tmp += 5;

if ($tmp == 17) {
echo "17 ans";
}

if ( $tmp == 20 ){ echo "20 ans";}
?>
```

# Partie 1 - Ecriture et standard

## Les mauvais exemples

# Partie 1 - Ecriture et standard

## Les mauvais exemples

```php
<?php
$ma_var="toto";
  $maVar2="tutu";
$tmp     ="15";

if($ma_var==$maVar2 )
    $tmp+=2;
    $tmp += 5;

if ($tmp == 17) {
echo "17 ans";
}

if ( $tmp == 20 ){ echo "20 ans";}
?>
```

# Partie 1 - Ecriture et standard

## Les mauvais exemples

```php
<?php
$ma_var="toto";
  $maVar2="tutu";
$tmp     ="15";

if($ma_var==$maVar2 )
    $tmp+=2;
    $tmp += 5;

if ($tmp === 17) {
echo "17 ans";
}

if ( $tmp == 20 ){ echo "20 ans";}
?>
```

```php
<?php
$prenom1 = "toto";
$prenom2 = "tutu";
$age = 15;

if ($prenom1 === $prenom2) {
    $age += 2;
}
$age += 5;

if ($age === 17) {
    echo "17 ans";
}
if ($age === 20) {
    echo "20 ans";
}
?>
```

## PSR-2 par le php-fig

www.php-fig.org/psr/psr-2/

- Code compréhensible par tous
- Maintenance plus simple
- Moins d'erreurs de syntaxe
- Partage et collaboration possible
- Professionnel

## PSR-2 par le php-fig

www.php-fig.org/psr/psr-2/

## Petit conseil de dernière minute

**Pratiquer avec des projets perso**

# FIN DE LA 1ère PARTIE

```php
<?php echo "Merci de votre attention !"; ?>
```