CDUI2 PHP

Sylvain PHILIP
contact@sphilip.com

INTRODUCTION À PHP

PRÉSENTATION DE PHP : UTILISATION

- Génération de contenu (HTML, images, pdf…)
- Traitement de données (formulaires)
- Manipulation de fichiers
- Gestion de bases de données (ex : MYSQL)

page1 .php

PRÉSENTATION DE PHP : PREMIER SCRIPT

Ouvrir un éditeur et taper le texte suivant :

```
<?php echo "Hello World !" ?>
```

- Enregistrer le fichier *page1.php*
- Dans un terminal, se positionner dans le répertoire où se situe le fichier.
- Démarrer le serveur web intégré à PHP : php -S localhost:8080
- Dans le navigateur, taper l'adresse suivante : http://localhost:8080

page1 .php

PRÉSENTATION DE PHP : PREMIER SCRIPT

Modifier le fichier page1.php :

```
<?php $titre = "Hello World!"; ?>
<!DOCTYPE html>
<html lang="fr">
<head>
  <title><?php echo $titre; ?></title>
</head>

<body>
  <h1><?php echo $titre; ?></h1>
  <?php echo date("d/m/Y H:i"); ?>
</body>

</html>
```

Dans le navigateur, actualiser l'adresse suivante : http://localhost:8080

SYNTAXE DU PHP

LES BALISES

- Le code PHP est encadré par des balises **<?php** et **?>**. Tout le code PHP doit être écrit entre ces balises. Chaque instruction se termine par un **;**

```php
<?php
echo "<h1>Hello World</h1>" ;
echo date ("d/m/Y") ;
?>
```

```php
<h1>
<?php echo "Hello World"; ?>
</h1>
<p>
<?php echo date ("d/m/Y"); ?>
</p>
```

PROGRAMMATION EN PHP

- Syntaxe : Les commentaires
  - Ils sont utilisés pour expliquer le code et **ne sont pas exécutés** par l'interpréteur. Ils peuvent être sur une seule ligne avec ***//*** ou sur plusieurs lignes entre ***/**** et ****/***.

```
<?php
// Ceci est un commentaire sur une ligne
/*
Ceci est un commentaire
sur deux lignes
*/
?>
```

LES VARIABLES

- Les variables en PHP commencent par le symbole **$** suivi d'un nom. Elles peuvent contenir des lettres, des chiffres et le caractère souligné (_). Les noms de variables sont sensibles à la casse. Le symbole **=** est l'opérateur d'assignation.

```php
<?php
$age = 25;
$prenom = "Alexandre";
?>
```

TYPES DE DONNÉES

- Chaînes de caractères (string)
- Nombres entiers (integer)
- Nombres à virgule flottante (float)
- Booléens (boolean)
- Tableaux (array)
- Objets

```php
<?php
"Alexandre";  // string
25; // integer
0.54;              // float
[1, 2, 3, 4];     // array
["un", "deux"]; // array
true; // boolean
false; // boolean
?>
```

OPÉRATIONS MATHÉMATIQUES

- PHP prend en charge les opérations mathématiques de base :
  - l'addition (+)
  - la soustraction (-)
  - la multiplication (*)
  - la division (/)

```php
<?php
echo 10 + 5;
echo 22 - 3;
echo 11 * 17;
echo (10 – 5) *  3;
?>
```

CONDITIONS

Une condition renvoie toujours un booléen (vrai ou faux)

==	équivalent
!=	non équivalent
===	strictement équivalent
!==	strictement différent
<	inférieur
>	supérieur
>=	supérieur ou égal
<=	inférieur ou égal
isset	test si une variable existe

```php
<?php
$age = 25;
$salaire = 2500;
$note = 8;
$prenom = "Jean";
$langage = "PHP" ;

$age >= 18; // true
$salaire > 2000; // true
$prenom == "Jean"; // true
$langage === "PHP"; // true
$note >= 10; // false
$note === 8; // true
$note === "8"; // false
$note == "8"; // true
isset($note); // true
isset($ecole); // false
```

OPÉRATEURS LOGIQUES

OU: A **||** B

A	B	résultat
true	true	true
false	true	true
false	false	false

ET : A **&&** B

A	B	résultat
true	true	true
false	true	false
false	false	false

```php
<?php
$age = 25;
$salaire = 2500;

$age > 40 && $salaire < 3000; // false
$age > 40 || $salaire < 3000; // true
$age < 40 && $salaire < 3000; // true
$age > 40 || $salaire > 3000; // false
?>
```

STRUCTURES CONDITIONNELLES

Les structures conditionnelles permettent d'exécuter du code en fonction d'une condition donnée.

```
if (condition est vraie) {
    … instructions
}
elseif (autre condition est vrai) {
    … instructions
}
else {
    … instructions
}
```

```php
<?php
$age = 25;

if ($age > 18) {
    echo "Vous êtes majeur!";
}
elseif ($age == 18) {
   echo "Vous êtes tout juste majeur!";
}
else {
    echo "Vous êtes mineur!";
}
?>
```

TP

page2.php

Exercice : Salutations en fonction de l'heure

1) Obtenez l'heure actuelle en utilisant la fonction **date("H")** de PHP.
2) Utilisez une structure conditionnelle (par exemple, **if**, **elseif**, **else**) pour déterminer la partie de la journée en fonction de l'heure (matin, après-midi, soir).
3) En fonction de la partie de la journée, affichez un message de salutation approprié.

Assurez-vous que votre code fonctionne correctement et que les messages de salutation correspondent à la partie de la journée en cours. Vous pouvez personnaliser les messages comme bon vous semble.

PROPOSITION

page2.php

```php
<?php
$heure = date("H");

if ($heure >= 5 && $heure < 12) {
    echo "Bonjour ! C'est le matin.";
} elseif ($heure >= 12 && $heure < 18) {
    echo "Bonjour ! C'est l'après-midi.";
} else {
    echo "Bonsoir ! C'est le soir.";
}
?>
```

Formater le résultat dans un document HTML

PROPOSITION HTML 1

page2.php

```php
<?php
$heure = date("H");
$message = "";

if ($heure >= 5 && $heure < 12) {
    $message = "Bonjour ! C'est le matin.";
} elseif ($heure >= 12 && $heure < 18) {
    $message = "Bonjour ! C'est l'après-midi.";
} else {
    $message = "Bonsoir ! C'est le soir.";
}
?>
<!DOCTYPE html>
<html>
  <head><title><?php echo $message ?></title></head>
  <body><h1><?php echo $message ?></h1></body>
</html>
```

PROPOSITION HTML 2

page2.php

```
<?php $heure = date("H"); ?>
<!DOCTYPE html>
<html>
  <head></head>
  <body>
    <h1>
    <?php if ($heure >= 5 && $heure < 12): ?>

    Bonjour! c'est le matin

    <?php elseif ($heure >= 12 && $heure < 18): ?>

    Bonjour! c'est l'après-midi

    <?php else: ?>

    Bonsoir ! c'est le soir

    <?php endif ?>
    </h1>
  </body>
</html>
```

LES BOUCLES

Les boucles permettent d'exécuter du code plusieurs fois jusqu'à ce qu'une condition soit remplie. La boucle **while** :

```php
<?php
$compteur = 0;

while ($compteur < 10) {
    echo "Tour de manège numéro $compteur";
    $compteur =  $compteur + 1;
}

echo "Le manège est terminé !";
?>
```

LA BOUCLE FOR

- Syntaxe : **for (initialisation; condition ; instruction) {}**

```php
<?php
for ($compteur = 0; $compteur < 10; $compteur++) {
    echo "Tour de manège numéro $compteur";
}

echo "Le manège est terminé !";
?>
```

LES TABLEAUX

Les tableaux en PHP permettent de stocker plusieurs valeurs dans une seule variable. Il y a deux types de tableaux :

- Tableau indexé (Indexé par un numéro)
- Tableau associatif (Indexé par une clé de type chaîne de caractères)

LES TABLEAUX INDEXÉS

Les tableaux indexés sont numérotés. **Ils commencent par l'index 0**. On peut parcourir tous leurs éléments avec l'instruction **foreach**.

```php
<?php
// Déclaration
$fruits = ["pomme", "poire", "orange"] ;

// Accéder à un élément du tableau
echo $fruits[1] ; // Affiche pomme

// parcourir tous les éléments
foreach ($fruits as $fruit) {
    echo "<p>$fruit</p>" ;
} ?>
```

LES TABLEAUX ASSOCIATIFS

Les tableaux associatifs sont indexés par une chaîne de caractères. On peut parcourir tous leurs éléments avec l'instruction **foreach**.

```php
<?php
// Déclaration
$proprietes = [ "couleur" => "rouge", "taille" => "20mm"] ;

// Accéder à un élément du tableau
echo $proprietes["couleur"] ; // rouge

// parcourir tous les éléments
foreach ($proprietes as $cle => $valeur) {
    echo "<p>$cle : $valeur</p>" ;
} ?>
```

LES FONCTIONS

Les fonctions en PHP permettent de regrouper un ensemble d'instructions pour les réutiliser plus tard. Elles peuvent prendre des paramètres en entrée et retourner une valeur en sortie.

```php
<?php
function addition ($a, $b)
{
    return $a + $b;
}

echo addition (10, 5); // 15
echo addition (111, 20); // 131
?>
```

```php
<?php
function ditBonjour ($nom)
{
    echo "Bonjour $nom";
}

ditBonjour("Jean"); // Bonjour Jean
ditBonjour("Pierre"); // Bonjour Pierre
?>
```

LES FONCTIONS NATIVES

Ce sont les fonctions directement accessibles avec le langage PHP

```php
<?php
echo strlen("Hello"); // 5
echo substr("Hello", 0, 3); // Hel
echo date("Y"); // 2023
```

https://www.php.net/manual/fr/

- substr
- strlen
- date

PORTÉE DES VARIABLES

- En programmation, la portée d'une variable désigne l'étendue de sa visibilité. Une variable peut avoir une portée **globale** ou **locale**. Une variable globale est accessible à partir de n'importe quel endroit du programme, tandis qu'une variable locale est limitée à un bloc de code spécifique (généralement une fonction).

PORTÉE DES VARIABLES

```php
<?php
$mois = date("m") ; // Portée globale

function afficheDate ()
{
    global mois ; // Récupération de la variable globale $mois
    echo "Mois numéro : $mois" ;

    $heure = date("H") ; // Portée locale
    // $heure ne sera pas accessible en dehors de la fonction afficheDate ()

    echo $heure ;
}

afficheDate() ;

echo $heure ; // Erreur $heure non définie
?>
```

SUPERGLOBALES PHP

En PHP, il existe plusieurs variables superglobales qui sont accessibles depuis n'importe quel contexte, à l'intérieur ou à l'extérieur des fonctions. Ces variables sont :

- **$_SERVER** : Un tableau associatif contenant des informations sur l'environnement d'exécution du script, comme les chemins des fichiers, les informations d'en-tête HTTP, etc.
- **$_GET** : Un tableau associatif contenant les données envoyées par le biais de la méthode GET.
- **$_POST** : Un tableau associatif contenant les données envoyées par le biais de la méthode POST.
- **$_FILES** : contient les informations sur les fichiers téléchargés.
- **$_COOKIE** : Un tableau associatif de variables, passé au script courant, via des cookies HTTP.

DÉVELOPPER EN PHP

ENVIRONNEMENTS DE DÉVELOPPEMENT

- Utilisation du serveur Web intégré à PHP :

```
$ : php -S localhost:8080 -t /repertoire/racine
```

Permet de créer un serveur web local accessible via l'adresse http://localhost:8080 . Il servira les fichiers se situant dans le dossier /repertoire/racine

- Installation de la suite Apache, Mysql, PHP (AMP).
  - MACOS : MAMP
  - Windows : WAMP
  - Ubuntu : `$ : apt install apache2 mysql-server php`

LOGS D'ERREURS ET DÉBOGAGE

- Mettre au maximum le rapport d'erreurs

```php
<?php error_reporting (E_ALL) ?>
```

- Afficher les erreurs

```php
<?php ini_set("display_errors", 1); ?>
```

- Afficher le type et le contenu de n'importe quelle variable

```php
<?php var_dump($var); ?>
```

- Afficher le contenu d'un tableau

```php
<?php var_dump($var); ?>
```

TRANSMISSION DE DONNÉES

TRANSMISSION DE DONNÉES

- Des données peuvent être transmises en clair dans l'URI. c'est la méthode HTTP **GET**
- Pour ce faire il faut ajouter un **?** après le nom du script suivi des paramètres sous la forme **clé=valeur** séparer par des **&**. Ex :
  - page.php**?marque=asus&format=micro**
  - https://www.google.com/search**?q=php**

EXEMPLE MÉTHODE GET

- page.php**?marque=asus&format=micro**
- Nous récupérons en PHP les variables **marque** et **format** en utilisant la variable superglobale **$_GET** (qui est un tableau associatif)

```php
<?php
echo $_GET["marque"] ; // asus
echo $_GET["format"] ; // micro
?>
```

Astuce : Utiliser print_r pour afficher le contenu d'un tableau :

```php
<?php print_r ($_GET) ?>
```

```
Array
(
    [marque] => asus
    [format] => micro
)
```

MÉTHODE POST

- Des données peuvent être transmises de façon transparente dans corps de la requête. c'est la méthode HTTP **POST**
- Un moyen pour poster des données est de créer un formulaire HTML ayant pour attributs : **method="post"** et **action="mon_script.php"**
- L'attribut **name** des champs du formulaire servira à nommer la variable à transmettre.

MÉTHODE POST : FORMULAIRE HTML

form.html

```html
<form method="post" action="traitement.php">
  <input type="text" name="nom" placeholder="Votre nom">
  <input type="text" name="age" placeholder="Votre age">
  <input type="submit" value="Envoyer">
</form>
```

Nous récupérons en PHP la variable age grâce à la variable superglobale **$_POST** :

traitement.php

```php
<?php
print_r ($_POST) ; // Pour déboguer
echo "Bonjour {$_POST['nom']}";
echo "Vous avez {$_POST['age']} ans";
?>
```

PROCESSUS DE TRAITEMENT DES DONNÉES TRANSMISES PAR MÉTHODES GET ET POST

1) On s’assure que l’on reçoit bien les données en utilisant par ex, la fonction native **isset()**

2) On nettoie les données afin qu’elles soient compatibles avec notre code.

3) On valide les données pour qu’elles s’inscrivent dans la logique de notre code

S'ASSURER DE RECEVOIR LES DONNÉES

```php
<?php
// Valeurs par défaut
$nom = "";
$age = 0;

if (isset($_POST["nom"])) {
    $nom = $_POST["nom"] ;
}

if (isset($_POST["age"])) {
    $age = $_POST["age"] ;
}
?>
```

NETTOYAGE DES DONNÉES

- Si l’on doit récupérer une variable contenant du texte, il faut s’assurer qu’elle ne contienne pas de caractères spéciaux comme des balises HTML.
- Si l’on doit récupérer une variable contenant un nombre, il faut s’assurer qu’elle contienne bien un nombre.

NETTOYAGE DES DONNÉES

```php
<?php
// Valeurs par défaut
$nom = "";
$age = 0;

if (isset($_POST["nom"])) {
    $nom = $_POST["nom"] ;
    // On s'assure que $nom ne contient pas de caractères spéciaux
    $nom = htmlspecialchars($_POST['nom']);
}

if (isset($_POST["age"])) {
    $age = $_POST["age"] ;
    // On s'assure que $age est un nombre
    $age = intval($age); // intval force $age à être un nombre
}
?>
```

VALIDATION DES DONNÉES

```php
<?php
// Valeurs par défaut
$nom = "";
$age = 0;

/*
... Nettoyage des données
*/

if ($age < 0 || $age > 121) {
        echo "Erreur : Votre âge n'est pas valide. L'âge doit être compris entre 0 et 121";
}
elseif ($nom == "") {
      echo "Erreur : Votre nom n'est pas renseigné";
} else {
     echo "Bonjour $nom, vous avez $age ans";
}
?>
```

TP

- Créer un formulaire puis afficher son résultat
  - Le formulaire consistera en 3 champs :
    - Civilité (Mr, Mme, Mlle)
    - Nom
    - Prénom
  - Une fois le formulaire envoyé, un message affichera : `Vos informations : {civilite} {nom} {prenom}`
  - S’assurer que la valeur de civilité soit (Mr, Mme, Mlle) et pas autre chose.
  - S’assurer que nom et prénom ne contiennent pas de code HTML.
  - S’assurer que nom et prénom contiennent bien une valeur.
  - Informer l’utilisateur de ses éventuelles erreurs de saisies.

SYNTHÈSE SUR L'EXPLOITATION DES DONNÉES TRANSMISES PAR L'UTILISATEUR

- Méthode GET : Récupération d'une ressource qui peut être filtrée selon des paramètres visibles dans l'URL.
- Méthode POST : Envoi de données de la part de l'utilisateur.
- Avec les deux méthodes, s'assurer de bien recevoir les données.
- Ne pas faire confiance aux données transmises (les nettoyer avant de les exploiter)

INCLUSION DE SCRIPTS

INCLUSION DE SCRIPTS

- Permet d'inclure du code provenant d'autres fichiers dans un script PHP. Avantages :
  - Évite les répétitions de code
  - Facilite la maintenance
- Syntaxe

```
<?php require "chemin/vers/un-fichier.php"; ?>
```

```
<?php include "chemin/vers/un-fichier.php"; ?>
```

DIFFÉRENCE ENTRE INCLUDE ET REQUIRE

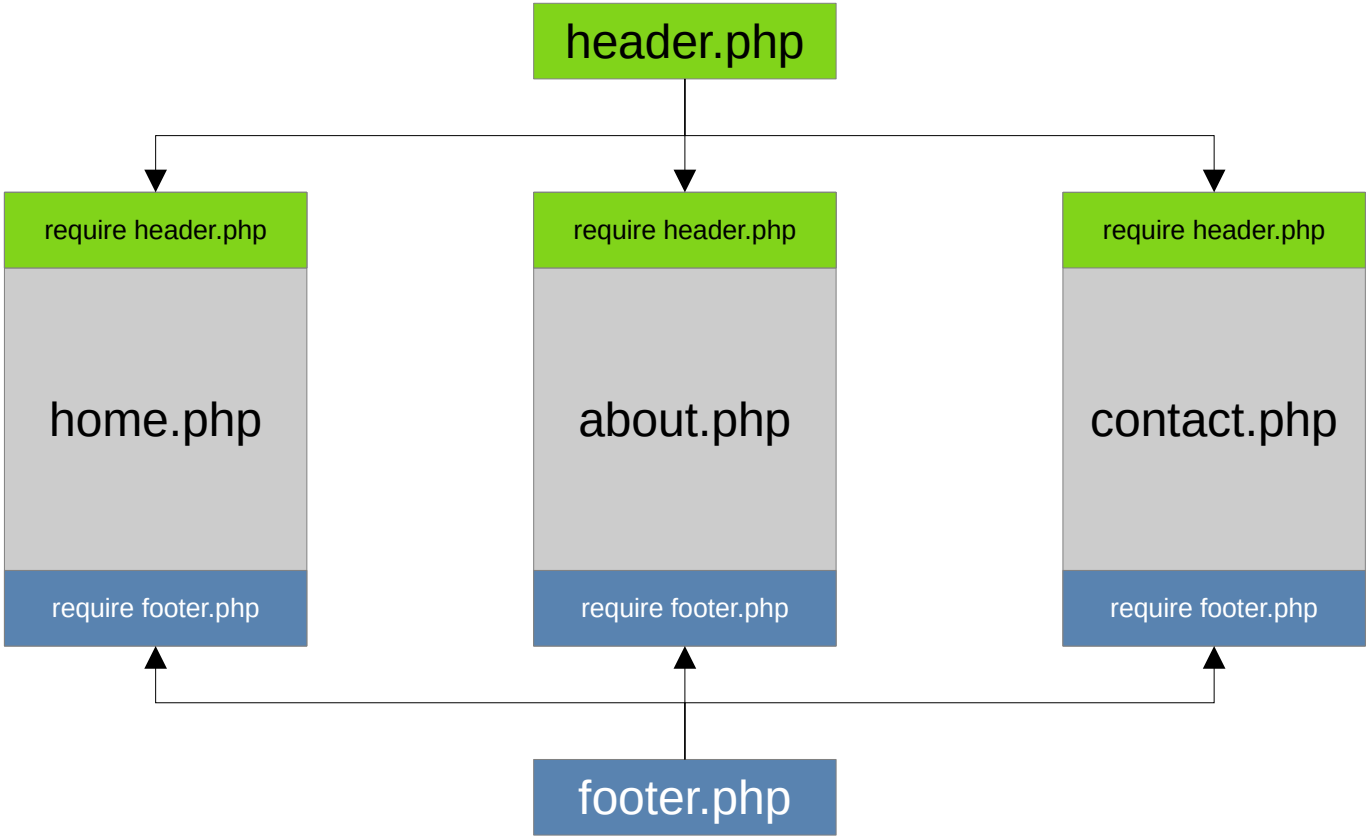
- **require** : Si le fichier n'existe pas, cela génère une **erreur fatale** de PHP (Le script arrête son exécution).
- **include** : Si le fichier n'existe pas, cela génère un **avertissement** de PHP (Le script continue)
- Lequel choisir : il est préférable d'utiliser **require** afin d'être assuré que le fichier sera inclus.

REQUIRE_ONCE INCLUDE_ONCE

- Même comportement que *require* et *include* à la différence que le fichier ne sera inclus qu'une seule fois (Lorsque l'on veut inclure des déclarations de fonction par exemple, car PHP n'autorise pas de multiples déclarations de la même fonction). C'est principalement ce type d'inclusion qui est utilisé dans Wordpress.

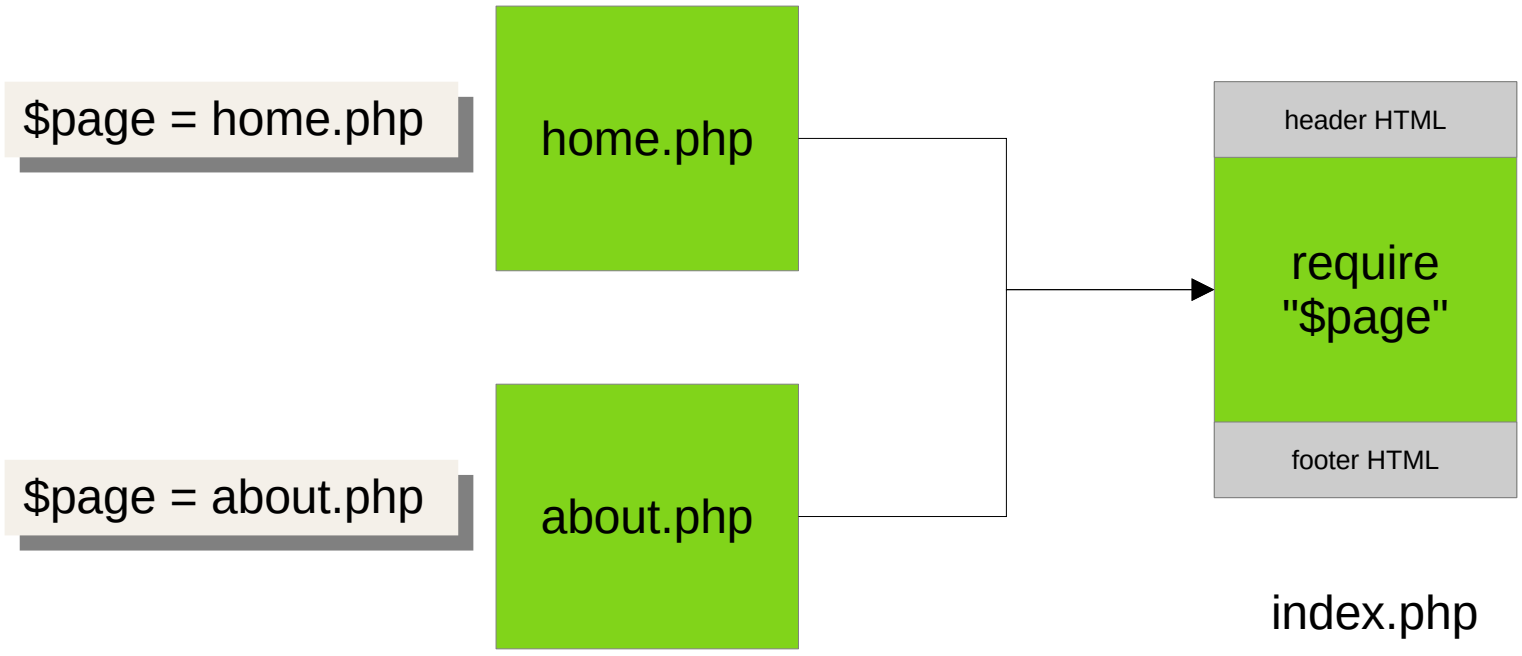
INCLUSIONS STATIQUES

La page à inclure est spécifiée directement dans le code



INCLUSIONS DYNAMIQUES

La page à inclure est choisie dynamiquement



INCLUSION ET SÉCURITÉ

- Lorsque l'inclusion est dynamique, veiller à contrôler la valeur de la variable afin de ne pas inclure n'importe quel fichier.

```php
<?php
/*
Exemple avec faille de sécurité
*/

$page = $_GET["page"] ;
require $page ;
?>
```

```php
<?php
/* Exemple sans faille de sécurité */
if ($_GET["page"] == "home") {$page = "home.php" ; }
elseif ($_GET["page"] == "about") {$page = "about.php" ; }
elseif ($_GET["page"] == "contact") {$page = "contact.php" ; }
else { $page = "not-found.php";}

if (file_exists ($page)) {
    require $page ;
}
?>
```

TP

inclusions/index.php

- Créer un mini site PHP de 3 pages : *Accueil, À propos, Contact*. chaque page sera incluse dynamiquement en récupérant le paramètre *page* dans l'url (méthode GET). Toutes les requêtes seront gérées par le fichier index.php

TEMPORISER LA SORTIE

- Sans temporisation, l'instruction **echo** ou du code HTML en dehors des balises PHP <?php ?> envoie immédiatement le texte à afficher au navigateur.
- Il est possible de temporiser (retarder) la sortie du texte en l'envoyant temporairement dans un tampon de sortie (Output Buffer)
- La fonction **ob_start()** sert à démarrer la temporisation dans le tampon de sortie.
- La fonction **ob_get_contents()** permet de récupérer le contenu du tampon de sortie.
- La fonction **ob_end_clean()** arrête la temporisation.

temporisation.php

TEMPORISER LA SORTIE : EXEMPLE

```php
<?php ob_start(); // Démarre la temporisation ?>
<?php // Ce qui se trouve au dessous ne sera pas affiché de suite ?>

<h1><?php echo "Hello World !";?></h1>

<p>Ceci est le contenu de ma page</p>

<?php $contenu = ob_get_contents(); ?>

<?php ob_end_clean(); // Termine la temporisation ?>

<!DOCTYPE html>
<html lang="fr">
<head>
  <title>Temporisation</title>
</head>

<body>
  <?php echo $contenu ?>
</body>

</html>
```

TP

- En reprenant le code du mini site, dans le fichier index.php, temporiser la sortie des pages à inclure avant la balise <html> puis récupérer le contenu des pages dans une variable $contenu. Cette variable sera ensuite affichée dans la balise <body>.

SESSION ET IDENTIFICATION

SESSION EN PHP

- Un cookie de session en PHP est un cookie qui est stocké sur le serveur et qui est associé à une session spécifique. Il est généralement utilisé pour stocker des informations spécifiques à une session, telles que l'identifiant de session, le nom d'utilisateur et d'autres informations. Les cookies de session sont supprimés lorsque l'utilisateur ferme le navigateur.
- Ne pas confondre avec les cookies du navigateur. Ces derniers sont stockés avec leurs données sur le poste du client. Dans le cas d’un cookie de session, seul l’identifiant de session est stocké côté client.

EXPLOITATION D'UNE SESSION EN PHP

- On démarre la session avec la fonction **session_start()**
- On récupère les données de session dans la super globale **$_SESSION**
- On détruit les données de session avec la fonction **session_destroy()**

SESSION : EXEMPLE SIMPLE

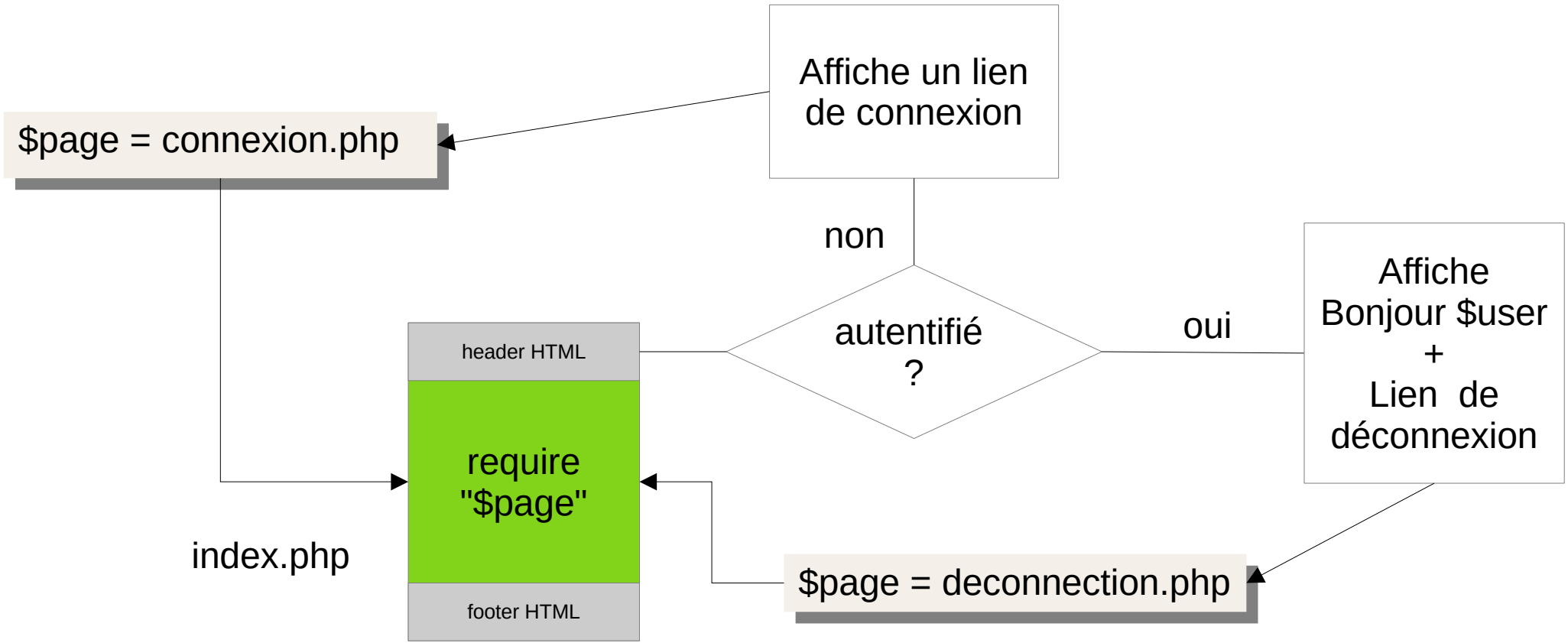
```php
<?php

session_start(); // Démarrage de la session

if (!isset($_SESSION['count'])) {
  // Si $_SESSION['count'] n'existe pas
  $_SESSION['count'] = 0; // On Crée  $_SESSION['count'] en lui assignant 0
} else {
  // Sinon on incrémente de 1 $_SESSION['count']
  $_SESSION['count']++;
}

echo $_SESSION['count'] ;
?>
```

INTÉGRATION DE L'AUTHENTIFICATION GRÂCE À LA SESSION



INDEX.PHP

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

// ...
?>
...

<?php if (isset($_SESSION["utilisateur"])) : ?>
Bonjour <?php echo $_SESSION["utilisateur"]["nom"] ?> <br>
<a href="index.php?page=deconnexion">Déconnexion</a>
<?php else : ?>
<a href="index.php?page=connexion">Connexion</a>
<?php endif ?>
```

CONNEXION.PHP

```php
<?php
if (isset($_POST["pass"])) {
    if ( $_POST["pass"]) == "sphilip") {
        $utilisateur = ["nom" => "Sylvain Philip"];
        $_SESSION["utilisateur"] = $utilisateur;
        header ("Location: index.php"); // Redirige sur la page d'accueil.
    }

    echo "<p>Échec d'authentification !</p>";
}
?>
<form method="post" action="index.php?page=connexion">
<input type="password" name="pass">
<input type="submit" value="Connexion">
</form>
```

DECONNEXION.PHP

```php
<?php
session_destroy(); // Destruction des données de la session
header ("Location: index.php"); // Redirige sur la page d'accueil.
?>
```

PHP ET LES BASES DE DONNÉES

BASES DE DONNÉES INTRO

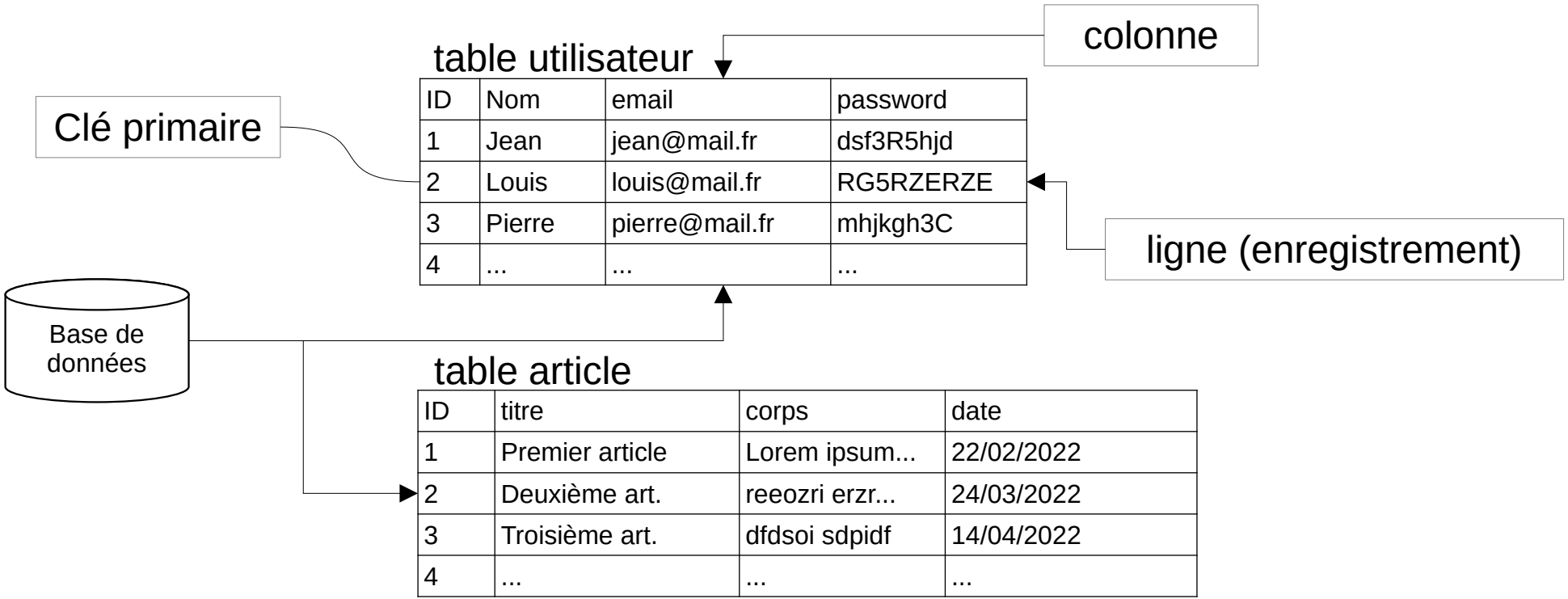
- Les bases de données servent à stocker et à organiser de façon structurée des données.
- Elles permettent de gérer, rechercher, récupérer et mettre à jour ces données de manière efficace.
- Elles sont utilisées dans les entreprises, les administrations, les sites Web, les banques.

SGBD (SYSTÈME DE GESTION DE BASES DE DONNÉES)

- Il existe une multitude de SGBD : (Oracle, Access, Mysql, Postgresql) tous accessibles via PHP.
- MYSQL est le plus couramment utilisé en PHP (Wordpress l'utilise)
- Une alternative est SQLITE, qui ne nécessite pas de connexion à un serveur.

ORGANISATION D'UNE BDD

Une base de données est organisée en tables. Chaque table à un certain nombre de colonnes et peut contenir plusieurs lignes (correspondant aux enregistrements). Chaque enregistrement peut être retrouvé à partir d'une clé primaire (L'identifiant).



ID	Nom	email	password
1	Jean	jean@mail.fr	dsf3R5hjd
2	Louis	louis@mail.fr	RG5RZERZE
3	Pierre	pierre@mail.fr	mhjkgh3C
4	...	...	...

ID	titre	corps	date
1	Premier article	Lorem ipsum...	22/02/2022
2	Deuxième art.	reeozri erzr...	24/03/2022
3	Troisième art.	dfdsoi sdpidf	14/04/2022
4	...	...	...

LE LANGAGE SQL

- Les SGBD utilisent un langage pour la gestion des BDD : Le **SQL**, ou **S**tructured **Q**uery **L**anguage
- **SQL** est un langage de programmation utilisé pour communiquer avec les bases de données relationnelles. Il permet de gérer et manipuler les données stockées dans une base de données.
- Vous écrivez des requêtes SQL pour interroger la base de données et récupérer des informations spécifiques.

REQUÊTES COURANTES

- **SELECT** : permet de récupérer des données spécifiques depuis une table ou plusieurs tables.
- **INSERT** : permet d'ajouter de nouvelles lignes de données dans une table.
- **UPDATE** : permet de mettre à jour les données existantes dans une table.
- **DELETE** : permet de supprimer des lignes spécifiques d'une table.
- **CREATE** : permet de créer une nouvelle table dans la base de données.
- **ALTER** : permet de modifier la structure d'une table existante, comme ajouter ou supprimer des colonnes.

EXEMPLES

```
CREATE TABLE utilisateur (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  nom VARCHAR(40) NOT NULL DEFAULT '',
  email VARCHAR(40) NOT NULL DEFAULT '',
  password VARCHAR(20) NOT NULL DEFAULT ''
);

INSERT INTO utilisateur (nom,email,password) VALUES ('Jean', 'jean@mail.fr', 'dfgsdfg');

UPDATE utilisateur SET password='dfsdfspo' WHERE email = 'jean@mail.fr';

SELECT * FROM utilisateur;

DELETE FROM utilisateur WHERE email = 'jean@mail.fr';
```

GESTION BDD EN PHP

- La classe PDO permet de se connecter à une BDD et d'effectuer des requêtes.

```php
<?php
// Connexion BDD sqlite
$db = new PDO("sqlite:/fichier.sqlite");
// ou Connexion MYSQL :
$db = new PDO("mysql:mysql:host=localhost;dbname=testdb", "username", "password");
?>
```

GESTION BDD EN PHP

```php
<?php
// Connexion BDD sqlite
$db = new PDO("sqlite:fichier.sqlite");

// On utilise query pour une requête SELECT
$res = $db->query("SELECT * FROM utilisateur");

while ($row = $res->fetch()) {
    echo $row['nom'] . ' ' . $row['email'];
}

// On utilise exec pour les autres requêtes

$count = $db->exec("UPDATE utilisateur SET password='dfo' WHERE email = 'jean@mail.fr';");

?>
```

EXERCICE

- En utilisant Sqlite, créer une table article contenant 5 colonnes :
  - id : clé primaire
  - titre
  - intro
  - contenu
  - date
- Ajouter quelques articles dans la table.
- Créer une nouvelle page articles dans le mini-site PHP
- Afficher les articles de la base en php selon ce modèle :

```
<h2><a href="index.php?page=article&id={id}">{titre}</a></h2>
<p>{date}</p>
<hr>
```

- Lorsqu'on clique sur le lien, afficher le lien de l'article dans une page en php selon ce modèle :

```
<h1>{titre}</h2>
<p>{date}</p>
<div>{contenu}</div>
```

ANNEXES

DIRECTIVES PHP

- PHP peut-être configuré par le biais de directives définies dans un fichier (php.ini)
- Il n’est pas possible de modifier directement ce fichier si l’on n’est pas administrateur du système.
- En tant qu’utilisateur, nous pouvons placer certaines directives dans un fichier *.user.ini*
- Ou bien les définir directement en php :

  <?php ini_set("nom_directive", "valeur") ; ?>

DIRECTIVES COURANTES

- **display_errors** : On / Off. Cette directive permet de spécifier si les erreurs doivent être affichées à l'écran ou non.
- **max_execution_time** : Cette directive permet de définir la durée maximale (en secondes) pendant laquelle un script PHP peut s'exécuter avant d'être interrompu.
- **memory_limit** : Cette directive permet de définir la quantité maximale de mémoire qu'un script PHP peut utiliser. (ex "128M" pour 128 méga octets)
- **upload_max_filesize** : Cette directive permet de spécifier la taille maximale des fichiers téléchargés. (ex "30M" pour 30 méga octets)
- **post_max_size** : Cette directive permet de spécifier la taille maximale des données envoyées via la méthode POST. (ex "30M" pour 30 méga octets)

RESSOURCES SUR LE WEB

- Manuel PHP
  https://www.php.net/manual/fr/
- Introduction au PHP
- Cours PHP en ligne
- Parts de marché de PHP
  https://kinsta.com/php-market-share/