PHP, DÉVELOPPER UN SITE WEB DYNAMIQUE

SOMMAIRE

OBJECTIFS PÉDAGOGIQUES

Maîtriser la syntaxe PHP en vue de développer de sites Web dynamiques.
Traiter des formulaires Web.
Gérer des sessions utilisateurs.
Accéder aux données d'une base de données MySQL.
Créer dynamiquement des images et des graphismes.

PRÉREQUIS

Connaissances de base du langage HTML et d'au moins un langage de programmation.
Connaissance de l'ordinateur et de son système d'exploitation.

NÉCESSAIRE POUR DÉVELOPPER UN SITE WEB EN PHP

Un éditeur de code (Microsoft Visual StudioCode par exemple).
Une plateforme de type WAMP ou MAMP.

INTRODUCTION

1 - L'architecture du Web

Ce que l'on appelle la toile, le Web en anglais, est une partie d'Internet en fait.

Le Web repose sur une architecture, notamment le protocole HTTP et des "passerelles communes" que l'on nomme CGI pour Common Gateway Interface.

Cela permet beaucoup d'interactivité sur les sites Web, tant du côté du navigateur que du côté du serveur. Cela se traduit par des "scripts", qui sont des fichiers de code, exécutés soit côté client soit côté serveur.

PHP est un langage dit "back-end" car il s'exécute côté serveur et intègre donc une CGI également.

Cette CGI, connu sous le nom PHP-CGI, permet une interaction avec le monde extérieur au serveur.

En général, vous entendrez parler de sites web dits "dynamiques", ce que permet le langage PHP !

2 - Qu'est-ce que PHP ?

PHP est un langage de programmation informatique, qui permet donc de créer des sites web dynamiques. Il est dit "interprété" et "back-end".

PHP offre plusieurs approches de programmation, soit “procédurale” soit “orientée objet”.

Ces approches s’appellent des paradigmes.

Le paradigme impératif ou “procédural” implique un code par procédure, c’est à dire que les différents blocs de code sont interprétés les uns à la suite des autres.

Le paradigme orienté objet ou “OO” implique que les différents blocs de code sont séparés dans différents fichiers, encapsulés dans des “objets”.

Concrètement, PHP nous donne la possibilité de communiquer avec une base de données, donc d’afficher des valeurs différentes sur la même page HTML en fonction de qui est connecté par exemple !

3 - Historique de PHP

Créé en **1994**, PHP n’est à la base qu’une librairie de fonctionnalités écrite en langage C et n’est donc pas un langage de programmation à proprement parlé.

Le code source de PHP a été publié la première fois en **1995**, par Rasmus Lerdorf, un jeune développeur groenlandais, qui voulait savoir combien de personnes visitent son CV en ligne !

En **1996**, Rasmus publie une version 2.0 de PHP, qui signifiait alors Personal Home Page – Form Interpretor.

Un interpréteur de formulaire, voilà qui allait donner des idées à deux jeunes étudiants israéliens.

Andi Gutmans et Zeev Suraski, recherchent pour un projet de fin d’étude une technologie permettant d’être interprétée côté serveur. C’est alors qu’ils découvrent PHP – FI 2.0.

Très satisfaits de cette nouvelle librairie, ils décident en **1997** de faire évoluer cette librairie. Ils passent une année entière à réécrire le cœur de PHP-FI 2.0.

En **1998**, nos deux étudiants publient une version 3.0, transformant la librairie en langage ! C’est un peu une révolution, et en profitent pour changer le nom en PHP :Hypertext Preprocessor.

En parallèle, une mascotte est créée la même année par l’artiste Vincent Pontier a.k.a ElRoubio.

Début des années **2000**, Andi et Zeev publient PHP 4.0, dont ils ont réécrit le moteur interne de PHP et l’appellent le Zend Engine.

Vingt ans plus tard, en 2020, la version 8.0 a été publié, apportant son lot d’ajout de fonctionnalités majeures.

Découvrons cela dans le prochain chapitre !

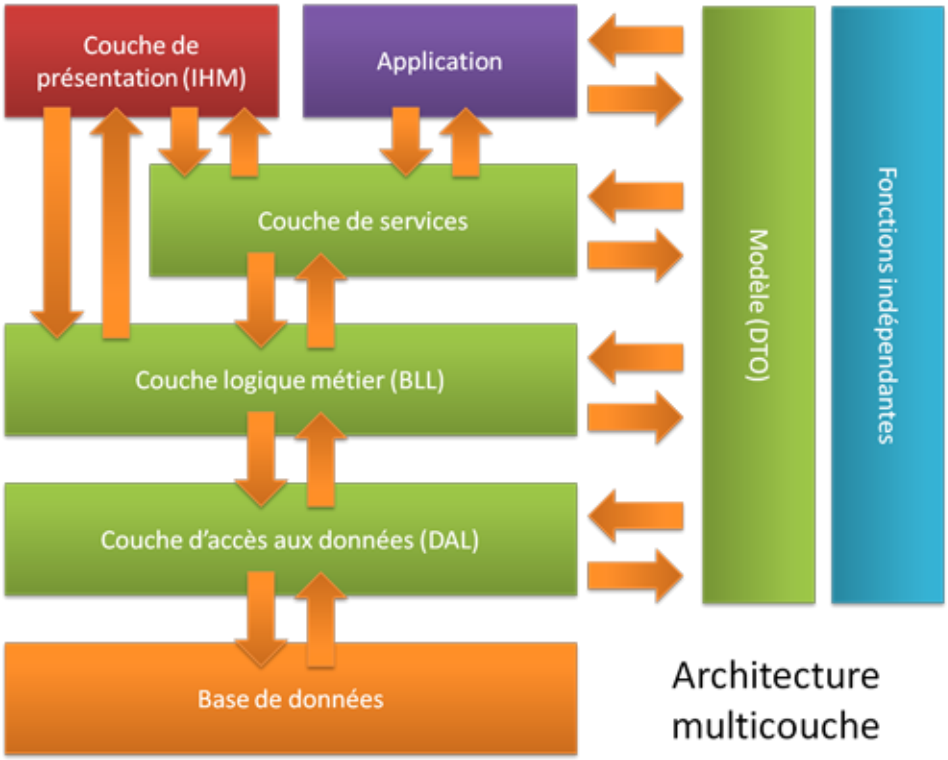
4 - Les différences entre PHP 4, PHP 5 et PHP 7

Version PHP	Description
4.0	Sécurité renforcée et Intégration du nouveau moteur Zend Engine.
5.0	Zend Engine 2
5.3	Support des “namespace” - Fonctions anonymes – Ajout de PHP-FPM (FastCGI Process Manager)
5.4	Ajout des “traits”
5.5	Ajout d’une API de hash – OPcache devient standard
5.6	Possibilité de télécharger des fichiers de 2Go+
7.0	PHP NG remplace Zend Engine – Typage des valeurs de retour pour les fonctions + Ajout de l’opérateur coalescent null “??”
7.1	Multiple catch possible – Ajout du type de valeur “nullable” et “void”
7.4	Typage des propriétés de classes
8.0	Compilation “Just-in-time"

5 - Notions d'architecture multicouches et introduction aux principes MVC.

L’architecture multicouches permet une répartition des différentes tâches d’une application Web.

Cette répartition des tâches permet de découper le code en plusieurs “couches”, qui ont chacune un rôle bien précis. Cela allège les opérations à effectuer pour chaque couche, augmente la maintenabilité de l’application Web ainsi que son potentiel d’évolution.



Architecture multicouche

Le **MVC** respecte cette architecture multicouche, c'est que l'on appelle un *patron de conception*, ou *design pattern* en anglais.

Les principes du **MVC** reposent sur une conception tripartite, les Modèles, les Contrôleurs et les Vues.

Le code se décompose alors comme suit :

- **Model** : représente une table en base de données
- **View** : représente une page HTML
- **Controller** : représente la logique de développement, soit le code des fonctions concernant un Model.

Cette organisation de conception se retrouve beaucoup dans la programmation orientée objet (POO).

Le **MVC** est implémenté dans les frameworks modernes tels que Symfony, Laravel ou Angular.

6 - Présentation du site Web e-commerce utilisé.

Voir dossier fourni "Boutique".

Travaux pratiques

Voir annexe "Travaux-Pratiques.pdf"

Revue des balises principales HTML et des commandes de style.
Introduction à la feuille de style de l'application exemple.

Automatisation d'une page Web

1 - Les principes du client-serveur

Le protocole ou environnement client–serveur désigne un mode de transaction entre programmes et processus.

D'un côté est représenté le client (navigateur) et de l'autre côté le serveur.

Le client envoie des requêtes au serveur, et le serveur répond au client.

Tout le monde connait la fameuse erreur 404, c'est une réponse du serveur lorsqu'il ne trouve pas la page HTML demandée lors de la requête !

2 - Premiers éléments du langage

Dans tous les langages de programmation dont PHP, vous retrouverez des notions communes. Découvrons-les ensembles :

- Les variables, notées avec le symbole $ en PHP et dont la valeur peut varier.
- Les constantes, notées en majuscule en PHP et dont la valeur est constante.
- Les fonctions, qui permettent de faire une action comme un calcul ou un upload de fichier.
- Les tableaux, appelés *array* et qui peuvent contenir plusieurs valeurs.
- Les boucles, qui permettent de faire des itérations de code.
- Les conditions, qui permettent d'exécuter du code si une condition donnée est respectée.
- Les opérateurs, qui permettent plusieurs actions comme comparer, calculer, assigner des valeurs.
- Les classes, utilisées en orienté objet, représente un objet avec ses propriétés et propres fonctions.

Ces éléments nous offrent tous les moyens syntaxiques pour coder dans le langage PHP !

3 - Intégration de HTML dans un fichier PHP

Le langage PHP offre la possibilité d'intégrer du code HTML dans un fichier PHP (exemple.php). Par contre l'inverse n'est pas possible.

Dans la capture d'écran suivante, examinons comment cela se traduit :

exemple.php

```php
<?php

echo "Je suis du code PHP";

?>

<h1>Je suis du code HTML dans un fichier PHP</h1>
```

La première ligne est une balise d'ouverture, qui indique que la suite est du code PHP.

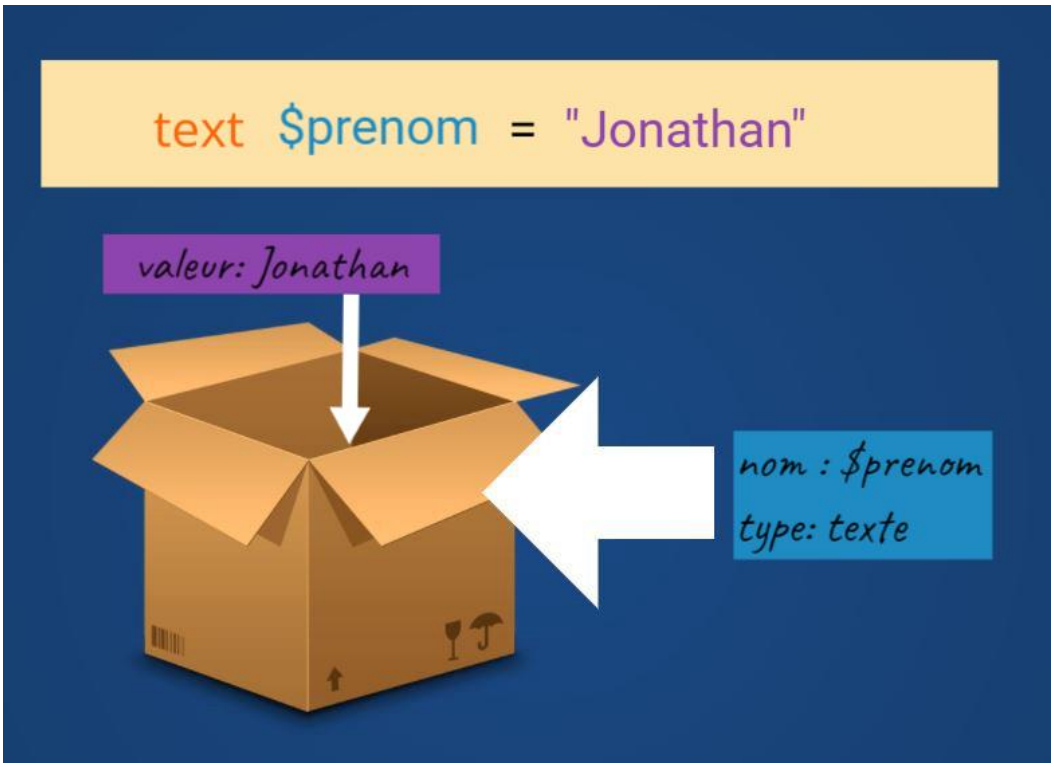
La ligne suivante est une instruction PHP, demandant à afficher le texte en jaune.

Ensuite la balise fermante de PHP indique que le bloc de code PHP se ferme à cet endroit.

La balise <h1> est une balise HTML qui sera interprétée et affichera le texte entre les deux balises.

4 - Variables et fonctions PHP

Le concept des variables est plutôt simple. Imaginez un carton, sur lequel on peut écrire un nom (pour l'identifier) et ce qu'il contient. À l'intérieur de ce carton, on peut y mettre une seule chose, une *valeur.*



Le concept de fonction est un peu plus délicat à cerner. Pour rendre cela plus simple, nous allons schématiser comme cela :

Cela se traduit comme suit en code PHP :

```php
# DÉCLARATION
function multiplierPar3($entree)
{
    $sortie = $entree * 3;
    return $sortie;
}

# APPEL
echo multiplierPar3( entree: 10);
```

Une exécution de fonction se fait en deux étapes :

- Une définition
- Un appel

Lors de la définition, qui est le premier bloc, il y a plusieurs parties qui composent la définition :

- Un nom
- Un ou des paramètres en entrée
- Un corps

L'appel se fait par le nom de la fonction et si elle en attend, ses paramètres.

Deux catégories de fonctions existent, les fonctions prédéfinies et les fonctions utilisateurs.

Il existe plusieurs fonctions prédéfinies dans PHP, comme des fonctions pour manipuler du texte, des nombres, des fichiers, des tableaux (array), des variables, de débogage et bien d'autres encore.

Pour plus de détails, voici le lien vers la documentation officielle :
https://www.php.net/manual/fr/indexes.functions.php

On reconnaît aisément les fonctions grâce à leurs parenthèses à la fin de leur nom.

5 - Librairies

Une librairie, ou *bibliothèque*, est un ensemble de variables et fonctions, souvent répartis en plusieurs fichiers.

Cela permet de rendre du code réutilisable. Comme pour une connexion à une base de données et sa gestion.

Vu que nous utilisons PHP dans ce but, de dialoguer avec une BDD, nous sommes amenés à le faire à chaque projet. Il serait inutile et chronophage de le coder à chaque fois, alors le code permettant de faire ce travail est codé une seule fois, puis est inclut dans les projets qui nécessitent une gestion de BDD.

En PHP, nous retrouvons aussi le nom de *dépendance* pour nommer une librairie.

6 - Variable serveur

Comme nous l'avons vu plus haut, PHP est un langage côté serveur. PHP nous permet donc d'avoir des informations de la part de ce dernier. Ces informations sont toutes stockées dans une seule variable mais une variable d'un genre particulier que l'on appelle superglobale.

Elle est globale, c'est à dire accessible depuis n'importe quelle partie de votre code, de manière globale.

Elle est super aussi, car elle contient plus de 150 valeurs !

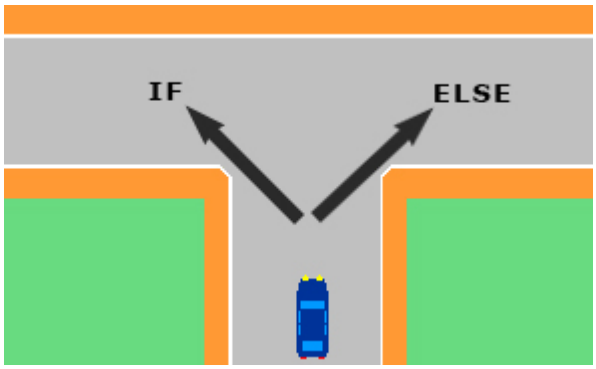
Cette superglobale s'appelle $_SERVER.

7 – Structures conditionnelles et boucles

PHP nous offre, comme dans la majorité des langages de programmation, différentes structures de code.

Les structures conditionnelles et les boucles en font partie.

Les premières permettent un contrôle du flux (du code). Imaginez une route et une voiture :



Il y a deux chemins possibles, à gauche et à droite. La route est le code et la voiture le flux. Si la voiture remplie une condition, alors elle tournera à gauche, sinon elle tournera à droite.

Appliquons cette représentation graphique en code PHP :

```php
$couleur = "";

if ($couleur === 'vert') {
    echo "$couleur est vrai";
}
else {
    echo "$couleur n'est pas vert";
}
```

Le résultat est conditionnel, soit l'exécution du code dans la partie du bloc *if*, soit du code de l'autre partie dans le bloc *else*.

Les conditions sont très utiles pour détecter si un utilisateur est un membre du site ou un administrateur par exemple. L'affichage pourra être différent et les fonctions disponibles également.

Les boucles quant à elles, permettent de répéter un bout de code autant de fois que souhaité. Pour éviter le comportement dit de "boucle infinie", elles se compose d'une expression conditionnelle également.

Découvrons comment une boucle se compose :

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

Dans cet exemple, l'instruction PHP dans le corps sera répété 10 fois et affichera une suite de chiffres partant de 1 et jusqu'à 10.

Cela est très pratique sur une page Web pour afficher une bloc HTML avec des variables PHP à l'intérieur.

On peut prendre l'exemple du site Pinterest, qui affiche plein de photos différentes, avec des catégories différentes, mais toujours dans une même structure HTML.

Il existe plusieurs structures conditionnelles dans PHP :

- La boucle ***for***
- La boucle ***foreach***
- La bouche ***while***
- La boucle ***do … while***
- La structure ***switch***

Travaux pratiques

Voir annexe "Travaux-Pratiques.pdf"

Réalisation de fonctions personnalisées.
Réalisation d'une librairie de fonctions.

Les formulaires simples

Dans le langage (de structure) HTML, il nous est possible de créer des formulaires. Ils sont très pratiques pour demander une réponse à l'utilisateur final, comme son prénom, son email ou son adresse postale.

PHP, quant à lui, va nous permettre de récupérer ces réponses saisies dans le formulaire en vue de les traiter - comme supprimer les espaces blancs potentiels ou passer en majuscule un nom de famille, voire même à hasher un mot de passe.

1 - Transmission d'informations

Lorsque vous êtes sur une page Web qui contient un formulaire, comme une recherche Google, une page d'inscription à Netflix ou encore une page de commande sur Amazon, il y a toujours un bouton pour soumettre le formulaire et s'il est validé alors votre recherche, inscription ou commande est effectuée.

Comment les données que vous avez saisis dans les différents champs du formulaire sont évaluées ?

Il y a plusieurs façons de transmettre des données du navigateur au serveur, et cela grâce aux requêtes qu'envoi le navigateur.

Souvenez-vous des superglobales de PHP, il y en a deux pour récupérer ces informations de formulaire.

On peut transmettre les données soit dans l'URL du navigateur, soit dans un tableau PHP. $_GET et $_POST sont les deux superglobales de PHP qui peuvent contenir les données d'un formulaire.

2 – Lecture et écriture de fichier

Vous devez savoir que dans un formulaire, on peut demander un fichier, comme une photo pour votre profil Facebook.

Ce fichier doit être lu avant d'être déplacé sur le serveur. PHP fournit toutes les fonctions pour faire cela. Les informations du fichier - comme son nom, son poids ou son extension - sont stockées dans une autre superglobale de PHP, nommée $_FILES.

C'est un tableau qui contient les informations nécessaires pour contrôler le fichier et l'uploader s'il est validé par notre code.

On peut donc détecter si c'est bien un fichier de type image pour une photo ou un pdf si c'est ce que l'on demande dans notre formulaire.

3 - Vérification des identifiants de connexion

Lors du traitement d'un formulaire de connexion, comme celui de votre boîte email, il faudra vérifier les identifiants saisis par l'utilisateur. Comme le mot de passe est hashé, on devra utiliser des fonctions de PHP pour déterminer si le mot de passe correspond bien à celui stocké en BDD.

4 – Redirection après validation du formulaire

Si tout se passe bien lors du traitement des données du formulaire, c'est à dire qu'aucune erreur n'est survenue, alors il faudra rediriger l'utilisateur sur une page HTML.

PHP offre cette possibilité, très pratique pour avoir une navigation fluide et cohérente dans votre site Web.

Les variables complexes : tableaux

Un array, ou tableau, est une variable qui contient plusieurs valeurs. Ces valeurs sont organisées sous forme de pair "clef" => "valeur".

Exemple 1 : *$prenoms = ["Léna", "Mina", "Angel", "Astrid", "Mélo"]*

Cet exemple est un tableau sans clés, mais si on regarde de plus près, il y a des chiffres correspondant à un prénom. On appelle cela un ***index***. Ils sont automatiquement générés et commencent toujours à 0.

0	1	2	3	4
Léna	Mina	Angel	Astrid	Mélo

Exemple 2 : $profil = [
"age" => 38,
"prenom" => "Mina",
"hobbie" => "Vélo"

]

Cet exemple montre un tableau avec des clés, ces clés ont un nom, comme “age” ou “hobbie” et leurs valeurs correspondantes. Ce sont des paires “clé/valeur” (*key/value* en anglais).

age	prenom	hobbie
38	Mina	Vélo

Les indexes peuvent être une chaîne de caractères ou un nombre. Si l'index est omis, un index numérique sera automatiquement généré, commençant à 0.

1 - Constructeur array()

Pour créer un tableau en PHP, il existe deux façons :

- En mettant les crochets en valeur de variable - $couleurs = []
- En faisant appel à une fonction native de PHP - $couleurs = array()

2 - Fonctions associées aux tableaux

PHP offre de nombreuses fonctions pour manipuler un tableau (array), par exemple en extraire une portion, réorganiser les indexes après la suppression d’une paire, inverser l’ordre défini, vérifier si une valeur existe dans le tableau, et plus encore.

La liste est vraiment longue, et complète, en suivant ce lien vous aurez la liste exhaustive : https://www.php.net/manual/fr/book.array.php

3 - Fonctions d'extraction

Il existe une fonction très pratique en PHP, qui permet d'extraire toutes les paires clés/valeurs d'un tableau d'un coup : extract()

À ce moment-là, une clé devient une variable. Reprenons l'exemple vu plus haut :

```
$profil = [
   "age" => 38,
   "prenom" => "Mina",
   "hobbie" => "Vélo"

] ;
```

Si on exécute la fonction extract() sur ce tableau, nous aurons ceci en résultat :

```
extract($profil) ;
```

Suite à cette instruction, nous obtiendrons de manière globale une variable $*age*, une $*prenom* et une $*hobbie*

4 - Fonctions de navigation dans un tableau

Pour pouvoir naviguer dans un tableau, il y a plusieurs fonctions natives dans PHP comme :

- next() : avance à l'index suivant
- prev() : recule à l'index précédent
- end() : avance au dernier index du tableau

On peut évidemment choisir nous-même l'index que l'on souhaite afficher, en le spécifiant de cette manière :

```
echo $profil[2] ;
```

On reconnaît facilement les tableaux PHP grâce à leurs crochets à la fin de leur nom.

Travaux pratiques

Voir annexe "Travaux-Pratiques.pdf"

Réalisation d'une fonction de création de liste déroulante.

Gestion des sessions utilisateurs

Une **session** est un système mis en œuvre dans le code PHP permettant de conserver sur le serveur, dans un fichier, des informations relatives à un internaute.

1 - Variables persistantes : cookies et session

En programmation informatique, il existe une notion que l'on appelle la ***persistance***. La persistance est le fait qu'une donnée ***survit*** dans le temps au programme qui l'a créé ou généré.

L'avantage d'une ***session*** c'est que les données seront enregistrées dans un fichier sur le **serveur** disponible et consultable par toutes les pages durant toute la navigation de l'internaute.

Par exemple, les données d'un panier ne sont pas conservées en BDD, car beaucoup de paniers n'aboutissent pas à une commande et donc à un paiement.

Chaque visiteur accédant à votre page web se voit assigner un identifiant unique, appelé "*identifiant de session*". Il peut être soit propagé dans l'URL, soit stocké dans un ***cookie***.

Pour démarrer une session, on utilise la fonction PHP ***session_start()***.
Cette fonction génère un identifiant, c'est ce qu'on appelle un "ID de session" ou PHPSESSID.
PHP transmet automatiquement cet ID de page en page (en utilisant généralement un cookie).
Ensuite les données de session sont disponibles dans une superglobale PHP, appelée $_SESSION.

Travailler avec des ***cookies*** revient à peu près à la même chose qu'avec des sessions, à quelques petites différences près que nous allons voir.
Un cookie, c'est un petit fichier que l'on enregistre sur **l'ordinateur** de l'utilisateur. Ce fichier contient du texte et permet de "retenir" des informations sur le visiteur.

Parfois, les cookies ont une mauvaise image. On fait souvent l'erreur de penser que les cookies sont « dangereux ».

Ce ne sont pas des virus, juste de petits fichiers texte qui permettent de retenir des informations.

Comme une variable, un cookie a un nom et une valeur.
Pour écrire un cookie, on utilise la fonction PHP ***setcookie()***.
Il y a en général trois paramètres à renseigner :

- Le nom du cookie
- La valeur du cookie
- La date d'expiration du cookie

Ensuite les données du cookie sont disponibles dans une superglobale PHP, appelée $_COOKIE.
Chaque cookie stocke généralement une information à la fois.

2 - Avantages et inconvénients des cookies et sessions

Un cookie identifie davantage une machine qu'un utilisateur. Si plusieurs utilisateurs partagent un même ordinateur familial sans ouvrir de sessions personnelles, ils partagent le même cookie.

Également, comme vu précédemment, un fichier de session est stocké sur le serveur, tandis qu'un fichier de cookie est stocké sur l'ordinateur de l'utilisateur.

L'inconvénient donc des cookies est que si un utilisateur se connecte à un même site Web, mais sur deux ordinateurs différents, alors les données stockées dans le premier ordinateur sont inaccessibles depuis le second.

3 - Limitations et précautions

Les utilisateurs peuvent supprimer ou éviter les cookies, ce qui limite l'utilisation des cookies en tant que développeur. Dans le cadre d'une navigation cross-device, les cookies ne permettent pas de suivre un utilisateur donné puisque chaque ordinateur, ou plutôt chaque navigateur, stocke un cookie différent.

Pour correctement paramétrer un fichier de cookie, il y aura quelques précautions à prendre pour le sécuriser :

- Activer l'option *httpOnly*
- Activer l'option *secure*

```php
<?php
// retenir l'email de la personne connectée pendant 1 an
setcookie(
    'LOGGED_USER',
    'utilisateur@exemple.com',
    [
        'expires' => time() + 365*24*3600,
        'secure' => true,
        'httponly' => true,
    ]
);
```

En écrivant les cookies de cette façon, vous diminuez le risque qu'un jour l'un de vos visiteurs puisse se faire voler le contenu d'un cookie à cause d'une faille XSS.

Il ne faudra JAMAIS placer le moindre code HTML **avant d'utiliser la fonction *setcookie()*** !

4 - Sérialisation des variables complexes

Une variable dite "complexe" n'est pas une variable difficile, mais dont sa valeur est complexe à représenter, par exemple comme un tableau – ou array.

La sérialisation est une technique pratique pour stocker ou passer des valeurs – ou données – en PHP.

Le principe de la sérialisation est le codage d'une information sous la forme d'une suite d'informations plus petites (dites atomiques) pour, par exemple, sa sauvegarde (persistance) ou son transport sur le réseau (proxy, HTTP…).
L'action inverse, visant à décoder cette suite pour créer une copie conforme de l'information d'origine, s'appelle la désérialisation.

Reprenons notre exemple de panier stocké en session.

Je souhaite stocker tout le contenu du panier - donc les identifiant de chaque produit, leur titre, leur prix, etc… - dans un fichier, comme un cookie par exemple. Voici son contenu :

```
array (size=2)
  458 =>
    array (size=5)
      'photo' => string 'photo458.jpg' (length=12)
      'reference' => string 'AB0001' (length=6)
      'titre' => string 'tee-shirt' (length=9)
      'quantite' => int 1
      'prix' => int 10
  137 =>
    array (size=5)
      'photo' => string 'photo137.jpg' (length=12)
      'reference' => string 'AB0054' (length=6)
      'titre' => string 'chemise' (length=7)
      'quantite' => int 1
      'prix' => int 45
```

Nous ne pourrons pas stocker la variable (`$_SESSION['panier']`) qui contient toutes ces données en soit. Une fonction en PHP existe et créer une chaîne de caractères contenant toutes les données, sans perdre leur structure ni leur type : ***serialize().***

Voici un exemple en code PHP, ainsi que son résultat :

```
48    $panier = serialize($_SESSION['panier']);
```

Si on affiche maintenant la variable $panier, cela donne :

```
a:2: {
       i:458; a:5 {
           s:5:"photo";s:12:"photo458.jpg";s:9:"reference";s:6:"AB0001";s:5:"titre";
           s:9:"tee-shirt";s:8:"quantite";i:1;s:4:"prix";i:10;

     }
     i:137; a:5{
           s:5:"photo";s:12:"photo137.jpg";s:9:"reference";s:6:"AB0054";s:5:"titre";
           s:7:"chemise";s:8:"quantite";i:1;s:4:"prix";i:45;

     }

}
```

C’est une simple chaîne de caractères - du point de vue d’un ordinateur – qui peut être enfin stockée.

Ensuite pour retrouver la variable d’origine et pouvoir interpréter ses données, la fonction ***unserialize()*** de PHP permet de faire l’action inverse :

```
58      unserialize($panier);
```

Ce qui nous permet d’avoir à nouveau un tableau PHP exploitable avec les fonctions natives de PHP – pour manipuler le tableau par exemple.

```
array (size=2)
  458 =>
    array (size=5)
      'photo' => string 'photo458.jpg' (length=12)
      'reference' => string 'AB0001' (length=6)
      'titre' => string 'tee-shirt' (length=9)
      'quantite' => int 1
      'prix' => int 10
  137 =>
    array (size=5)
      'photo' => string 'photo137.jpg' (length=12)
      'reference' => string 'AB0054' (length=6)
      'titre' => string 'chemise' (length=7)
      'quantite' => int 1
      'prix' => int 45
```

Travaux pratiques

Voir annexe “Travaux-Pratiques.pdf”

Réalisation d'un panier d'achat simple, version cookie et session.
Gestion des quantités commandées.

Utilisation d'une base de données MySQL

Une base de données (en anglais : *database*, abrégé DB) est utile dans le cadre d'un projet informatique pour conserver des informations – ou données - en mémoire. C’est la fameuse persistance !

A l'intérieur de la base de données, les informations sont classées, structurées et regroupées généralement par sujet (Produit, Catégorie, Commande, Stock, Utilisateur, etc..).

Lorsque nous postons un message sur un forum et que deux semaines plus tard le message est toujours présent, c'est parce que le message a été sauvegardé dans une base de données (en français : abrégé *BDD*).

Dans la grande majorité des cas une base de données est gérée par un logiciel moteur qui la manipule : un ***SGBD - ou*** **Système de Gestion de Base de Données**.

1 - Présentation de MySQL

Voici le lien de la documentation officielle MySQL : https://dev.mysql.com/doc/

MySQL fait partie des *Système de Gestion de Base de Données,* qui est un logiciel qui s'installe sur un disque dur pour pouvoir l'utiliser.

Il contient tout le nécessaire pour créer des bases de données, ajouter des données et les conserver dans le temps. MySQL permet d'utiliser le langage SQL, qui est un langage de requêtes pour les BDD.

SQL signifie Structured Query Langage, il nous permet d'échanger des informations avec la base de données. Une fois que les informations ont été enregistrées, il est important de pouvoir les gérer (ajout, modification, suppression, consultation).

Toutes ces actions de gestion et manipulation passeront par une requête SQL.

Pour nous permettre de faire tout cela sur notre propre ordinateur – dit développement en local - il existe des logiciels gratuits content :

- PHP
- SQL
- MySQL
- Et même un serveur HTTP – Apache

Un logiciel de ce type s'appelle une plateforme – de développement local – et il en existe une pour chaque système d'exploitation, *Windows, MacOS ou Linux,* et se nomme **Wamp, Mamp ou Lamp.**

Les bases de données sont incontournables dans tous les domaines... et même hors du web. Nous sommes tous enregistrés dans des Bases De Données sans forcément s'en préoccuper - par exemple : la sécurité sociale, votre banque, la CPAM, etc.

2 - Concepts fondamentaux : bases, tables, champs, enregistrements.

Avoir une base de données ne représente rien de très complexe.

Une ***base*** est un espace nommé, qui contiendra des tableaux – ou ***tables***. Chaque table aura un nom également et contiendra des colonnes – ou ***champs***. Chaque champ aura aussi un nom et contiendra une donnée – ou ***enregistrement***.

Voici un exemple, car une image vaut mille mots :

Vous pouvez observer que la base de données s'appelle ***entreprise*** et la table ***employés***.

Et la table en question, avec en haut en bleu le nom des champs et chaque ligne d'enregistrements :

id_employes	prenom	nom	sexe	service	date_embauche	salaire
1	Jean-Pierre	Laborde	m	direction	2010-12-09	4750
2	Clement	Gallet	m	commercial	2010-12-15	2300
3	Thomas	Winter	m	commercial	2011-05-03	3550
4	Chloe	Dubar	f	production	2011-09-05	1900
5	Elodie	Fellier	f	secretariat	2011-11-22	1600
6	Fabrice	Grand	m	comptabilite	2011-12-30	2900
7	Melanie	Collier	f	commercial	2012-01-08	3100
8	Laura	Blanchet	f	direction	2012-05-09	4500
9	Guillaume	Miller	m	commercial	2012-07-02	1900
10	Celine	Perrin	f	commercial	2012-09-10	2700

On s'aperçoit que cette table est relative à des employés d'une entreprise.

Nous pratiquerons évidemment du SQL dans notre projet PHP, car nous aurons une base de données pour conserver toutes les données nécessaires à notre site Web e-commerce.

Nous pourrons voir les base SQL, les tables SQL, les champs SQL et les enregistrements SQL.

3 - Fonctions PHP MySQL

PHP implémente plusieurs fonctions natives, prêtes à l'emploi. Cela nous permettra de créer une connexion à une BDD existante – et créée par nos soins – ainsi que d'ajouter des données, les modifier ou les supprimer, et les afficher. Cette opération s'appelle un ***CRUD*** – pour **C**reate **R**ead **U**pdate **D**elete.

Ce sont les 4 actions possibles sur les données.

En PHP une connexion à une BDD peut se faire avec la classe ***mysqli***, qui représente une connexion entre PHP et une base de données MySQL :

```php
$mysqli = new mysqli("localhost", "user", "password", "database", 3306) ;
```

Ou avec la classe ***PDO*** :

```php
$pdo = new PDO('mysql:host=localhost;dbname=test', $username, $password);
```

Une requête pourra être faite ensuite, grâce à par exemple la fonction ***query()*** :

```
$requete = $mysqli->query("SELECT prenom, nom, salaire, FROM employes");
```

La classe PDO possède la même fonction ***query()*** que la classe mysqli.

Voici deux liens vers la documentation qui liste toutes les fonctions des deux classes :

- Mysqli : https://www.php.net/manual/fr/class.mysqli.php
- PDO : https://www.php.net/manual/fr/class.pdo.php

4 - Introduction au langage SQL (sélection, modification, suppression)

Voici le lien de la documentation SQL : https://sql.sh/

Nous allons voir comment en langage SQL effectuer les quatre actions possibles, souvent appelé CRUD comme vu plus haut.

- Pour la création - ou insertion – de données, SQL utilise un mot-clé précis, **INSERT INTO … VALUES** :

```
INSERT INTO nom_table (nom_colonne_1, nom_colonne_2)
VALUES ('valeur1', 'valeur2') ;
```

Ce qui donnerait pour notre exemple de table employes :

```
INSERT INTO employes (id_employes, prenom, nom, sexe, service, date_embauche, salaire)
VALUES (350, 'Jean-pierre', 'Laborde', 'm', 'direction', '2010-12-09', 5000)
```

- Le mot-clé SQL pour modifier un enregistrement est **UPDATE** … **SET** :

```
UPDATE employes SET salaire = 2500 ;
```

Mais attention, en l'état la requête modifiera ***TOUS*** les enregistrements dans la colonne ***salaire*** ! Utilisez une condition avec le mot-clé **WHERE.**

```
UPDATE employes SET salaire = 2500 WHERE id_employes = 123 ;
```

- Pour lire les données enregistrées dans une table, SQL fournit le mot-clé **SELECT** :

```
SELECT * FROM employes ;
```

Cette requête **affichera *toutes*** les <u>données contenues</u> dans la table *employes*.

L'astérisque signifie ALL en anglais, ou TOUT en français.

On peut lire la requête comme ceci : AFFICHE tout DE LA TABLE employes

- Enfin pour supprimer un enregistrement – ou ligne d'une table SQL, on pourra utiliser le mot-clé **DELETE** :

```
DELETE FROM employes WHERE prenom = 'Jean-Pierre' ;
```

Cette requête supprimera de la table tous les employés dont le prénom est Jean-Pierre.

On préfèrera utiliser la colonne d'identifiant (id) pour supprimer une ligne précisément, ici `id_employes`.

La suppression telle quelle est très sensible car <u>les données sont irrémédiablement perdues !</u>

<u>5 - Traitement des résultats des requêtes</u>

Grâce aux fonctions natives de PHP, nous pourrons traiter le résultat de ces requêtes.

Tout d'abord nous devons créer une requête SQL, grâce à la fonction ***query()*** fournit par **PDO** :

```php
$query = $connexion_bdd->query("SELECT * FROM produit");
```

Cette fonction <u>ne retourne pas les résultats</u> directement, mais un nouvel objet de type **PDOStatement** !

Il y a une autre fonction à exécuter ensuite, que l'on **stocke** dans une **variable.** Cette fonction, fournit par le nouvel objet **PDOStatement** s'appelle ***fetch()*** :

```php
$resultats = $query->fetch();
```

Vérifions ce que contient la variable ***$resultats*** grâce à la fonction de debug ***var_dump()*** :

```php
var_dump($resultats);
```

Et ce que nous affiche le navigateur :

```
array (size=11)
  'id_produit' => string '1' (length=1)
  'reference' => string 'AB001' (length=5)
  'categorie' => string 'Pull' (length=4)
  'titre' => string 'Pull à capuche' (length=15)
  'description' => string 'Un pull chaud et stylé pour l'hiver' (length=36)
  'couleur' => string 'vert' (length=4)
  'taille' => string 'M' (length=1)
  'public' => string 'homme' (length=5)
  'photo' => string 'photo-naketano.jpg' (length=18)
  'prix' => string '80' (length=2)
  'stock' => string '11' (length=2)
```

Nous venons de récupérer toute une ligne d'enregistrements en BDD ! Nous avons même **stocké toutes ces données** dans une **variable PHP**, une variable de type *array* - soit un tableau.

Au cours du projet nous pratiquerons tout cela car nous aurons une base de données !

Travaux pratiques

Voir annexe "Travaux-Pratiques.pdf"

Création d'une base MySQL.
Remplissage de la base à partir d'une base texte.
Création de fiches produit à la volée par extraction des données de la base.

Les formulaires complexes

1 – Moteur de recherche : sélection et tris des résultats

Un moteur de recherche n'est ni plus ni moins qu'un formulaire, la plupart du temps en méthode 'GET' !

Il suffit de regarder dans l'URL après une recherche sur le moteur de Google, devinez qu'elle a été ma recherche dans cet exemple :

Ce formulaire est relié à la base de données de Google pour pouvoir me retourner les résultats en lien avec ma recherche.

Nous aurons donc un formulaire HTML relié à notre ***BDD entreprise*** (créée précédemment), ainsi qu'un traitement avec **PHP et SQL** pour retrouver en BDD les éléments concernés par la requête de recherche.

La fonction de tri peut se faire en même temps, ou alors être au choix de l'utilisateur, ou encore directement sur les colonnes des résultats, dans une table HTML, avec un simple clic de la souris, pour ordonner le tout suivant l'ordre choisi.

Travaux pratiques

Voir annexe "Travaux-Pratiques.pdf"

Réalisation d'un moteur de recherche : la sélection sur auteur, titre et héros donne une liste de liens sur les fiches produit correspondantes.
Implémentation multicouche.

Le graphisme en PHP

PHP peut aussi servir à créer et manipuler des images, dans un grand choix de formats, comme GIF, PNG, JPEG, WBMP et XPM. Et **PHP** peut même *générer directement des images* pour le navigateur, avec la bibliothèque **GD**.

1 – La librairie GD2

```php
// Récupération des datas de taille et MIME type de l'objet passé
$info = getimagesize($sourcePathName);

// Vérification et création de ressource en fonction du MIME type de l'objet passé
if ($info['mime'] === 'image/gif')
    $image = imagecreatefromgif($sourcePathName);

elseif ($info['mime'] === 'image/png')
    $image = imagecreatefrompng($sourcePathName);

// Récupération de la ressource créée ci-dessus et conversion de l'image source en .jpeg
imagejpeg($image, $sourcePathName, $quality);
```

https://www.php.net/manual/en/ref.image.php

https://libgd.github.io/pages/docs.html

Travaux pratiques

Voir annexe "Travaux-Pratiques.pdf"

Intégration des différents modules réalisés.
Affichage des images avec mention de Copyright.

Le Plan de cours

Plan de cours

01 Introduction et révision HTML/CSS

02 Fonctions PHP et réalisation librairie

03 MySQL et création d'une base SQL

04 Les formulaires simples et upload de fichiers

05 Les array et réalisation d'une liste HTML

06 Session PHP et réalisation d'un panier

07 Les formulaires complexes et moteur de recherche

Librairie GD2 et les images avec PHP

ORSYS formation

*Learning for success**