Cas Pratiques et Travaux Pratiques

INTRODUCTION

SOMMAIRE :

- Présentation de l'exemple utilisé durant la formation : le site de vente en ligne.

Pour être conforme aux recommandations HTML 5, un document doit avoir la structure "*Doctype*".

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>

</body>
</html>
```

Cette première page est écrite en HTML pur, et tous les visiteurs de votre site verront exactement le même contenu, quel que soit le moment de leur connexion. Le fichier peut avoir l'extension .html ou .htm car il ne contient que du code HTML.

Il existe deux types de balises HTML : le type "*inline*" et le type "*block*".
Le premier, le type inline, se comporte de manière à ce que les balises soient côte à côte, sur la même ligne.
Le type block, quant à lui, va se comporter de manière à ce que les éléments HTML soient les uns en dessous des autres.
Par exemple : une balise <a> est de type inline, tandis que la balise est de type block.

Dans ce premier cours pratique, nous allons revoir ensemble les principales balises HTML utilisées ainsi que découvrir le projet Boutique utilisé durant la formation.
Ouvrons le dossier fourni "**Boutique-projet**" dans VisualStudio Code et plaçons-nous dans le dossier TP.
Nous allons écrire nos premières balises dans un fichier que l'on créer et l'on nomme TP1-HTML_CSS.html

La feuille de style CSS

Les feuilles de style en cascade, généralement appelées CSS de l'anglais Cascading Style Sheets, forment un langage informatique qui décrit la présentation des documents HTML.

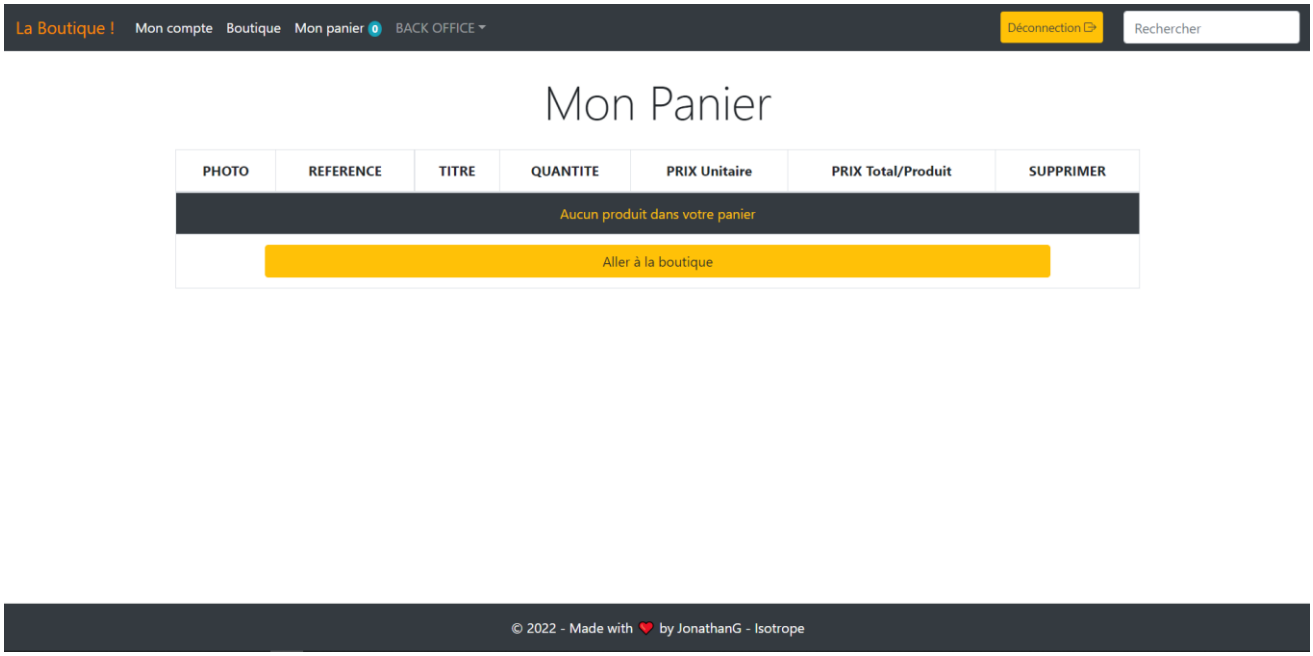
Connaissez-vous son rôle ? *Gérer la mise en forme de votre site.*

C'est lui qui vous permet de choisir la couleur de votre texte.
Lui qui vous permet de sélectionner la police utilisée sur votre site.
Lui encore qui permet de définir la taille du texte, les bordures, le fond…

Et aussi, c'est lui qui permet de faire la mise en page de votre site.
Vous pourrez dire : je veux que mon menu soit à gauche et occupe telle largeur, que l'en-tête de mon site soit calé en haut et qu'il soit toujours visible.

Voici un comparatif visuel de la page d'accueil sans CSS :

Et voici la même page HTML mais avec son style appliqué, avec donc CSS :



Il existe plusieurs moyens d'écrire du code CSS :

1. Dans un fichier CSS
2. Dans un fichier HTML
3. Dans l'attribut d'un fichier HTML

<u>CSS directement dans un fichier HTML</u>

```
<!doctype html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>

    <style>
        .ma-classe{
            color: green;
        }
    </style>

</head>
<body>

<div class="ma-classe">Tout ce texte est de couleur verte dans le navigateur</div>

</body>
</html>
```

CSS dans un attribut HTML

```html
<!doctype html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>

<div style="color: green">Tout ce texte est de couleur verte dans le navigateur</div>

</body>
</html>
```

Le seul avantage de ces deux techniques réside dans le fait qu'il est possible de tout écrire dans au sein du même fichier.
L'inconvénient c'est que ce style ne pourra pas être utilisé ailleurs, à moins de le réécrire à nouveau, ce qui peut être long et imprécis.

Nous privilégions donc une feuille de style CSS à part entière en plus de notre fichier HTML.

CSS dans un fichier .css externe

```html
<!doctype html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <link rel="stylesheet" href="style.css">

    <title>Document</title>
</head>
<body>

<div class="ma-classe">Tout ce texte est de couleur verte dans le navigateur</div>

</body>
</html>
```

Nous retrouvons dans le fichier *style.css* le même code que dans le premier exemple.

Travaux pratiques

Revue des balises principales HTML.
Introduction à la feuille de style CSS.

Automatisation d'une page Web

SOMMAIRE :

Comment intégrer du code PHP dans une page HTML ? En réalité la question est mal tournée, car c'est du HTML que l'on intègre à une page PHP.

Restons dans notre dossier TP2, et créons un nouveau fichier pour l'exemple, appelé ***php-et-html.php***.

Nous allons apporter un brin de dynamisme en affichant la date du jour en tête de page à l'aide du code PHP suivant :

```
1   <!doctype html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta name="viewport"
6             content="width=device-width, user-scalable=no, initial-scale=1.0">
7       <meta http-equiv="X-UA-Compatible" content="ie=edge">
8       <title>TP_2 : Intégration PHP et HTML</title>
9   </head>
10      <body>
11
12          <h1>Bonjour, la date du jour est : <?php echo date( format: 'd/m/Y'); ?></h1>
13
14      </body>
15  </html>
```

Ce qui donne comme résultat dans le navigateur :

Bonjour, la date du jour est : 08/11/2022

Lorsque vous voyez un mot-clé (qui n'est pas le vôtre), en anglais, suivi de parenthèses, c'est qu'il s'agit d'une fonction prédéfinie de PHP.

La fonction *date()* permet d'afficher la date avec les arguments suivants :

d : pour *day* (jour)
m : pour *month* (mois)
Y : pour *year* (année)

Le fait de mettre les arguments (d/m/Y) en minuscule ou majuscule à une incidence sur le résultat. Faites le test !

Tout comme notre langue, PHP possède plusieurs mots-clés et fonctions déjà existants dans le langage, nous pourrons les retrouver dans la documentation officielle de PHP : https://www.php.net/

Pour cette partie, nous coderons le reste de ce chapitre dans le dossier TP2. Allons créer un fichier appelé ***les-bases.php.***
Nous découvrirons dans ce fichier les premiers éléments du langage PHP.

Nous découvrirons la librairie GD2 un peu plus tard dans le programme, cela sera plus concret pour son utilisation.

Puisque PHP est un langage serveur, nous avons à disposition une sorte de variables qui contiennent des informations. Pour récupérer celles concernant le serveur, nous utiliserons cette variable : ***$_SERVER.***
On appelle ce genre de variables des SuperGlobales. Il en existe plusieurs, que nous découvrirons dans ce programme.

Pour voir les informations contenues dans cette superglobale *$_SERVER*, nous devrons utiliser une fonction native de débogage de PHP, *var_dump()*, car les superglobales sont en réalité des tableaux.

```php
var_dump($_SERVER);
```

Le résultat est conséquent car *$_SERVER* contient presque 200 lignes, soit autant de valeurs. C'est pour cela que l'on ne peut pas afficher son contenu avec un simple *echo.*

Travaux pratiques

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

Les formulaires simples

SOMMAIRE :

- Passage et transmission de variables.
- Lecture/écriture de fichier.
- Vérification de Login/mot de passe.
- Redirection.

Utiliser un formulaire HTML avec PHP permet d'échanger des données avec l'internaute. C'est-à-dire, lui envoyer des données, mais aussi en recevoir de sa part.

Autant dire qu'un site web sans formulaire n'existe pas !

Les formulaires reviennent très régulièrement sur les sites web, par exemple pour un formulaire de contact, un formulaire d'inscription en tant que membre, un formulaire de recherche, un formulaire de commande, un formulaire de paiement, un formulaire de satisfaction client et bien d'autres !

Comment récupérer les informations saisies dans un formulaire ?

Nous savons qu'avec le langage HTML il est possible de concevoir un formulaire. Avec CSS nous pourrons le styliser, mais il n'est pas nécessaire au bon fonctionnement de la transmission de données.

Concentrons-nous sur la transmission de données, mais avant regardons la balise HTML <form> :

```
<form action="#" method="get">
```

Il y a deux attributs obligatoires pour le bon fonctionnement du formulaire. Le premier est "***action***" qui permet de renseigner la page de destination lorsque le formulaire est soumis. Cela peut être une page de traitement, ou encore une page de résultats. La page en question correspond à un fichier, HTML ou PHP.

Le second attribut "***method***" correspond à la façon dont les données saisies dans les champs vont être transmis. Il en existe deux, **GET** et **POST**.
La méthode GET "injecte" les données du formulaire dans l'URL du navigateur. Regardons le formulaire en détails :

```
<form action="#" method="get">
    <label for="prenom">Quel est votre prénom ?</label>
    <input type="text" name="prenom" id="prenom">

    <input type="submit" value="Valider" style="background-color: limegreen;">
</form>
```

Et voici le formulaire dans le navigateur :

Si je tape mon prénom dans le champ, lorsque je soumets le formulaire, c'est-à-dire que je clique sur le bouton "Valider", dans l'URL on s'aperçoit que le prénom est injecté, ainsi que sa valeur. L'input est "*prenom*" et la donnée correspondante est "*Jonathan*" :

```
/formulaire_simple-et-tableaux.php?prenom=Jonathan
```

Nous pouvons maintenant "*regarder*" dans la **superglobale** qui correspond à cette méthode de transmission *GET*, pour cela faisons un dump de cette variable ***$_GET*** :

```php
var_dump($_GET);
```

Nous avons ce résultat à l'écran lorsqu'on rafraîchit la page web, et le résultat représente ce que ***contient*** le formulaire :

```
array (size=1)
  'prenom' => string 'Jonathan' (length=8)
```

Faisons ensemble la même procédure mais pour la seconde méthode, ***POST*** !

Découvrons maintenant comment avoir un champ pour "*uploader*" des fichiers, comment préparer le formulaire et comment *récupérer* ce fichier et le traiter grâce à PHP.

Nous devons avoir un input de type "file" dans notre formulaire HTML d'abord :

```html
<label for="photo">Choissisez votre photo ?</label>
<input type="file" id="photo" name="photo">
```

Imaginons que nous attendons une image de profil, les fichiers seront donc de type *.png* ou *.jpeg* par exemple.

En l'état, notre formulaire ne fonctionnera pas pour l'upload de fichier, la variable $_GET ne contient que le nom du fichier, or on souhaiterait vérifier le type de fichier, son poids ou encore son emplacement en vue de le déplacer dans un dossier approprié.

Modifions un peu notre balise ouvrante <form> pour ajouter un attribut, appelé "***enctype***" :

```
<form action="#" method="post" enctype="multipart/form-data">
```

Vous remarquerez sans doute que la méthode change aussi, nous devons être en méthode **POST** pour que cela fonctionne.

Rafraîchissons notre page web dans le navigateur et ajoutons un fichier dans le champ, puis faisons un *var_dump()* de la superglobale $_POST :

```
array (size=1)
  'prenom' => string 'Jonathan' (length=8)
```

Nous ne voyons pas le fichier ajouté, pourtant tout s'est bien passé !

PHP est "*organisé*" de façon à ne pas mélanger tout et n'importe quoi 😉. Il se trouve que nous avons à disposition une autre superglobale, appelée ***$_FILES*** en toute simplicité, et qui représente les fichiers chargés depuis un formulaire ! Regardons cela grâce à un *var_dump()* :

```
array (size=1)
  'photo' =>
    array (size=5)
      'name' => string 'image.png' (length=9)
      'type' => string 'image/png' (length=9)
      'tmp_name' => string 'C:\wamp64\tmp\phpA970.tmp' (length=25)
      'error' => int 0
      'size' => int 152000
```

Cette superglobale est aussi un ***array*** (tableau), un peu spécial car il contient un *autre array* ! On appelle cela un **tableau multi-dimensionnel** ou encore **tableau associatif**.

Nous avons enfin toutes les informations nécessaires pour manipuler le fichier. Listons-les :

- Le nom du fichier
- Le type du fichier
- Le dossier temporaire dans lequel le fichier se trouve
- Le code d'erreur : *0 signifie que tout est ok*
- La taille du fichier exprimée en octet : *Ici cela correspond à 152 Ko*

Il existe des fonctions natives pour cela dans le langage PHP, et la fonction move_uploaded_file() nous permet de *déplacer* le fichier dans un dossier prévu à cet effet, "***uploads"***.

En réalité cette fonction ***copie*** le fichier, c'est à dire qu'***elle lit le fichier et l'écrit ailleurs***, dans un dossier que l'on aura stipulé dans le code.
Créons maintenant un dossier nommé "uploads" dans notre TP3-Formulaire-simple.

Regardons cela de plus près :

```
move_uploaded_file($from, $to);
```

La fonction attend deux arguments, le dossier dans lequel se trouve le fichier actuellement, c'est un dossier temporaire, un peu comme une zone de transit. Nous avons déjà cette donnée, qui se trouve dans $_FILES, à l'index *'photo'* et dans la clé '*tmp_name*'.
Nous stockerons cette valeur dans une variable, ici appelée ***$from*** :

```php
$from = $_FILES['photo']['tmp_name'];
```

Pour le second argument attendu par la fonction *move_uploaded_file()*, que nous avons appelé ***$to***, nous allons composer cette variable ***$to*** avec plusieurs variables, à partir de toutes les informations et données à disposition :

```php
$dossierProjet = $_SERVER["DOCUMENT_ROOT"] . "/E-Boutique_PHP-master/";
$dossierUploads = "TP/TP3-Formulaire-simple/uploads/";
$nomPhoto = $_FILES['photo']['name'];
```

Nous pourrons donc avoir une variable ***$to*** composée de la <u>concaténation</u> de ces trois variables :

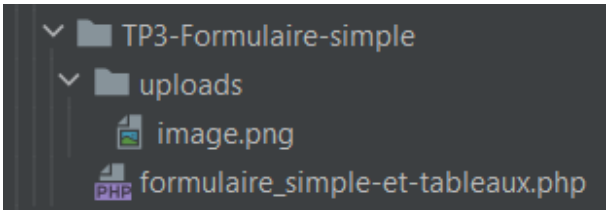
```php
$to = $dossierProjet . $dossierUploads . $nomPhoto;
```

<u>REMARQUE</u> : Les deux arguments attendus par la fonction *move_uploaded_file()* devrons être des chemins absolus, par exemple les variables ***$from et $to*** ont pour valeur :

$from = `'C:\wamp64\tmp\php609B.tmp'`

$to = `'C:/wamp64/www/E-Boutique_PHP-master/TP/TP3-Formulaire-simple/uploads/image.png'`

Essayons à présent de soumettre un fichier dans notre formulaire et observer le résultat :



La photo a bien été copié dans notre dossier ***uploads*** ! Félicitations, nous savons dorénavant comment uploader un fichier depuis un formulaire ✨

Découvrons maintenant le cas d'un formulaire d'inscription, avec un mot de passe !

Ajoutons à notre formulaire un nouveau champ pour cela, avec un input de type *'password'* :

```
<label for="mdp">Choisissez un mot de passe</label>
<input type="password" name="mdp" id="mdp">
```

Rafraîchissons notre page web pour écrire un mot de passe et soumettre le formulaire :

```
array (size=2)
  'prenom' => string 'Jonathan' (length=8)
  'mdp' => string 'toto' (length=4)
```

La valeur est dite en ***clair*** ! Et cela est un vrai problème car il est lisible directement par un humain. On doit pour cela créer un ***hash*** de cette valeur, ici *'toto'.*

Un hash est un mécanisme qui transforme une chaîne de caractères en une autre mais illisible par un humain. Le langage PHP nous fournit des fonctions pour exécuter ce mécanisme de *hashage*.

Faisons appel de la fonction ***password_hash()*** qui permet de créer une valeur à partir du mot de passe saisi dans le formulaire et '*dumpons*' le résultat :

```
$passwordHash = password_hash($_POST['mdp'], algo: PASSWORD_BCRYPT);
var_dump($passwordHash);
```

```
'$2y$10$KIwLRczqYhSuuxt8X7OXluIygGDRw5RInu03Fc3mGG1T/rXRrcDHq'
```

Cette fonction attend deux arguments, le premier étant le mot de passe en clair, le second est le type d'algorithme pour créer le hash. Il existe plusieurs algorithmes pour cela et PHP supporte les principaux (voir la doc : https://www.php.net/manual/en/function.password-hash.php).

En général, dans un contexte professionnel, c'est cette valeur que vous stockez en base de données. Cela se passe lors d'une inscription sur n'importe quel site web actuellement.

Maintenant supposons que nous avons une page de connexion et que nous souhaitons comparer le mot de passe saisi dans le formulaire de connexion avec la valeur stockée en BDD.

PHP fournit la fonction ***password_verify()*** qui permet, comme son nom l'indique en anglais, de vérifier un mot de passe. Cette fonction attend deux arguments, le mot de passe *en clair* et le mot de passe *hashé* :

```
$isPasswordOk = password_verify($_POST['mdp'], $passwordHash);
var_dump($isPasswordOk);
```

La réponse est : `boolean true`

Nous découvrirons un exemple plus proche de la réalité dans le dossier de projet fourni “Boutique-projet”.

Il reste une dernière étape pour boucler le processus de contrôle et validation de formulaire, c’est la redirection ! Lorsque vous cliquez sur le bouton ‘Valider’, il se passe le traitement, que nous avons codé, mais les fonctions de débug comme var_dump() n’ont pas vocation à rester dans le code final.

L’utilisateur final, lui, ne doit pas voir toutes ces étapes ! On va le rediriger vers une autre page de notre site pour le notifier du bon déroulement de son action, qui est dans notre exemple une inscription.

Pour cela nous allons créer un nouveau fichier nommé ***redirection.php***, à la racine du dossier *TP3*. Déclarons juste un doctype HTML et ajoutons un titre <h1> dans le <body> du document.

Au-dessus du doctype, nous ferons juste trois var_dump() en PHP :

```php
<?php
    var_dump($_GET);
    var_dump($_FILES);
    var_dump($_POST);
?>
```

Maintenant nous pouvons coder la ligne PHP pour rediriger vers ce fichier php. Retournons dans le fichier ***fichier-et-password.php*** et ajoutons cette ligne à la suite de notre code PHP :

```php
header( header: 'location: redirection.php');
```

Rechargeons notre page web avec le formulaire et validons-le après avoir renseigné tous les champs. Nous atterrissons sur notre nouvelle page et le titre <h1> s’affiche correctement, tout s’est correctement déroulé !

Nous allons continuer de manipuler des formulaires dans la suite du programme, avec les tableaux PHP, ou autrement appelés array !

Les variables complexes : tableaux

SOMMAIRE :

Comme nous l'avons vu, un tableau, ou array, est un type de variable. Ce type de variable permet de stocker plusieurs valeurs, qui sont organisées avec des indexes, ou clés, que l'on représente par paire de clé => valeur.

Créons un nouveau dossier **TP3-Les-Tableaux-PHP**, dans lequel on crée un fichier ***les-array.php*** pour nos exemples.

Pour déclarer un tableau, ou *array* et donc le créer en même temps, nous pouvons le faire de deux manières :

```php
$tableau = [] ;
$tableau = array() ;
```

Ces deux façons permettent de construire un tableau PHP, la première étant la plus courante. La seconde faisant appel au constructeur ***array()***, qui permet de faire la même chose que la première façon.

Les *array* se distingue par leurs crochets, comme déjà vu avec les superglobales $_GET et $_POST ! Ils se construisent donc avec une paire de crochets vide, si vous ne souhaitez pas ajouter de valeurs tout de suite.

PHP va nous fournir des fonctions prédéfinies pour manipuler les tableaux. Ces fonctions natives sont organisées en sous catégories, des fonctions pour manipuler les clés ou indexes des tableaux, d'autres pour les valeurs des clés, d'autres pour réorganiser l'ordre des paires 'clés/valeurs', encore d'autres pour vérifier si une clé existe dans un tableau, et la liste est longue encore !

Regardons la documentation officielle : https://www.php.net/manual/fr/book.array.php

Allons manipuler notre premier tableau, ou plutôt ***array*** !

Déclarons un premier *array* **$prenoms** dans notre fichier créé précédemment ***les-array.php*** :

```php
$prenoms = ["Léna", "Mina", "Angel", "Astrid", "Mélo"];
```

L'organisation est telle à l'intérieur des crochets que l'on peut facilement représenter ce schéma par un tableau :

0	1	2	3	4
Léna	Mina	Angel	Astrid	Melo

Le premier index est toujours le chiffre zéro ! Un array commence donc toujours à 0, et la valeur de cet index est 'Léna', et ainsi de suite dans l'ordre de déclaration faite juste avant. Ce tableau représente la variable $prenoms.

Maintenant, on souhaiterait observer le résultat de cet ajout. Si vous essayez de le faire comme pour une variable simple, à savoir avec l'instruction ***echo***, il y aura une erreur ! Elle est normale, car on ne peut demander d'afficher qu'une seule valeur à la fois avec *echo*, or un array contient plusieurs valeurs !

On ne pourra donc afficher qu'**un seul** des prénoms présents dans la liste ***$prenoms[]*** à la fois.

Pour faire cela, on doit sélectionner, entre les crochets, l'index correspondant au prénom. Par exemple pour afficher le prénom Mina, PHP demande de faire :

```php
echo $prenoms[1];
```

Cela affichera dans le navigateur le prénom Mina !

À présent, vous voulez très certainement savoir comment ajouter un nouveau prénom dans cette liste *$prenoms[].* On peut utiliser une fonction native, ***array_push()***. Entre les parenthèses on doit renseigner le tableau concerné, et la valeur à ajouter :

```php
array_push( &array: $prenoms,  ...values: 'Jonathan');
```

Ou alors utiliser une syntaxe – ou notation – raccourcie et qui fortement conseillée (pour des raisons d'optimisation des performances d'exécution du code), elle est même plus simple à écrire :

```php
$prenoms[] = "Jonathan";
```

On reprend notre variable array déjà définie $prenoms[] et on lui ajoute une nouvelle valeur avec le signe égal, ici de type *string* et de valeur *'Jonathan'*.

Vérifions maintenant si la valeur 'Angel' existe dans le tableau. PHP a dans son arsenal la fonction ***in_array()*** pour faire cela :

```php
var_dump(in_array( needle: 'Jonathan', $prenoms));
```

Notez que pour avoir un résultat visuel de cette fonction, on devra faire un *var_dump().*
Sans surprise, la réponse est ***true***, car j'ai ajouté cette valeur au tableau juste avant 😉.

Maintenant imaginons que vous souhaitez vérifier si une clé - ou index – existe :

```php
var_dump(array_key_exists( key: 5, $prenoms));
```

La réponse est *true*, vraie, car l'index numéro 5 existe bien dans le tableau *$prenom.* Tandis que si on demande à vérifier que l'index numéro 6 existe :

```php
var_dump(array_key_exists( key: 6, $prenoms));
```

La réponse est ***false***, fausse, car il n'y a pas d'index numéro 6 dans ce tableau.

Pour connaitre le nombre total de valeurs contenues dans un array, on peut utiliser la fonction ***count()***, native à PHP :

```php
count($prenoms);
```

Faite un *echo* pour afficher la valeur de retour de cette fonction, qui représente le total du tableau.

Sachez qu'un array, un tableau donc comme *$prenom[]*, peut prendre tout type de valeur. Créons une nouvelle variable $produit de type array, et ajoutons ces valeurs :

```php
$produit = ["teeshirt", "vert", 25, true];
```

Cela est valide pour PHP, et le code sera correctement exécuté, à savoir la variable $produit sera créée avec les valeurs données dans la déclaration. Mais on s'aperçoit que cela n'est pas très lisible, bien que l'on devine que '*25*' est certainement le prix du tee-shirt, ou '*vert*' la couleur, mais '*true*' on ne sait pas vraiment.

C'est pour cette raison que l'on peut renommer les indexes d'un tableau !
Observez cette nouvelle déclaration de notre variable $produit :

```php
$produit = [
  'nom' => 'teeshirt',
  'couleur' => 'vert',
  'prix' => 25,
  'en_stock' => true
];
```

Cela devient bien plus lisible et compréhensible pour nous si on déclare le tableau ainsi.
Vous comprenez peut-être mieux pourquoi on parle de paire 'clé => valeur'.

On peut utiliser maintenant une fonction de PHP qui permet d'extraire toutes les clés (donc la partie de gauche d'un array), sous forme de variable simple PHP, en ajoutant juste un symbole dollar $ devant le nom de la clé :

```php
extract( &array: $produit);
```

Essayons d'afficher la valeur de la clé couleur, en faisant comme si on avait déjà déclaré cette variable $couleur :

```php
echo $couleur;
```

Cela nous affiche bien la valeur 'vert' !

Excellent, nous avons vu ce qu'est une variable de type ***array*** et comment manipuler un peu cela, sans erreur. Vous vous demandez à présent comment naviguer dans un tableau, et nous avons déjà vu une façon, avec les crochets et le numéro de l'index ($prenoms[5]).
On peut faire appel à des fonctions native de PHP, comme ***array_key_first()*** qui retourne la première clé d'un tableau, ou ***array_key_last()*** qui fait l'inverse.

Néanmoins, il reste une étape cruciale à découvrir, c'est l'affichage de toutes les valeurs d'un coup. Pour cela on devra parcourir le tableau, à l'aide d'une boucle PHP ! Souvenez-vous, il nous restait une boucle à découvrir dans les bases de PHP, mais on devait découvrir les tableaux avant 😉

Cette boucle c'est foreach(), qui se traduit par 'pour chaque' en français. On parle de principe d'itération, de répétition si vous préférez.

```php
foreach ($produit as $value) {
    echo $value . "<br>" ; }
```

Cela permet de répéter l'instruction echo autant de fois qu'il y a de clés dans le tableau $produit.

Nous avons cela en affichage dans le navigateur, les quatre valeurs :

teeshirt
vert
25
1

Mais nous n'avons pas le nom des clés, alors pour cela on va modifier un peu notre boucle foreach() :

```php
foreach ($produit as $key => $value) {
    echo $key . " => " . $value . "<br>";
}
```

Le résultat est bien plus lisible car on sait à quoi correspond chaque valeur maintenant :

nom => teeshirt
couleur => vert
prix => 25
en_stock => 1

REMARQUE : À la clé '*en_stock*', on ne voit pas *true* mais *1*, c'est parce que le code PHP l'analyse comme cela. *False* est représenté par un *0*. Ce qui veut dire que ceux deux valeurs numériques valent tantôt *1* ou alors *true* !

Travaux pratiques

Réalisation d'une fonction de création de liste déroulante.

Gestion des sessions utilisateurs

SOMMAIRE :

Cela sert à plusieurs cas d'utilisation :

- La connexion d'un membre à un site web.
- Le panier d'un site ecommerce (vente en ligne).
- Les formulaires à plusieurs étapes.
- D'autres informations générales. Cela peut être marketing (derniers produits vus, suggestions de produits adaptés aux préférences de l'internaute selon sa navigation), etc.

Pour conclure, nous utiliserons les sessions pour sauvegarder des informations temporaires et garderons l'utilisation de base de données pour sauvegarder des informations durables.

Nous allons créer un fichier de session, dans lequel nous pourrons stocker des valeurs dites ***persistantes***.

Avant de coder cela, allons créer un fichier ***creer-manipuler-session.php*** dans le dossier **TP4**.

Les valeurs contenues dans un fichier de session se trouvent dans la superglobale $_SESSION.
C'est un array, donc nous le manipulerons comme nous avons vu dans le chapitre précédent 😊.

Faisons par curiosité un var_dump() de cette superglobale PHP. Nous avons un warning car le fichier n'existe pas encore, on doit le créer. Pour faire cela on va utiliser la fonction ***session_start()*** :

```php
session_start();
```

Refaisons un *var_dump($_SESSION)* maintenant. Plus d'erreur, mais un tableau vide !

Créons une première clé, on va la nommer 'panier' :

```php
$_SESSION['panier'] = [];
```

Maintenant nous allons créer la structure du panier, qui aura plusieurs clés :

```php
$_SESSION['panier'] = [

    'id_produit_425' => [
        'nom_produit' => 'teeshirt',
        'prix' => 25,
        'quantite' => 2
    ]
];
```

Nous allons nous exercer un peu sur la manipulation du tableau de session.

```
# Calculer le montant total de ce panier.
# Changer la valeur de la quantité à 1.
# Ajouter un nouveau produit nommé 'pull' au prix de 50 et à un seul exemplaire.
```

Découvrons quelques fonctions sur la session :

- session_name() : permet d'avoir le nom de la session
- session_save_path() : permet d'avoir le chemin absolu du dossier contenant le fichier
- session_module_name() : permet d'avoir le type de module, ici un fichier.

Pour supprimer un index de la session, il nous faudra utiliser la fonction native unset(), pour vider le panier par exemple :

```php
unset($_SESSION['panier']);
```

Maintenant que nous avons vu comment faire avec la session, nous allons voir comment faire avec un cookie.

Depuis quelques années, les sites web Européens ont l'obligation d'informer les internautes lorsque des informations liées à leur navigation et utilisation sont retenues dans un fichier cookie, sur leur ordinateur.
La loi européenne RGPD a vocation de renforcer la protection des données des utilisateurs.

Créons un nouveau fichier ***creer-manipuler-cookie.php*** dans le dossier **TP4.**

Pour créer un cookie avec PHP, vous avez la fonction native ***setcookie()***, qui prend plusieurs paramètres :

```php
setcookie('panier_id_produit', "425", time()+7200, '/', '', true, true);
```

Il y a une syntaxe alternative plus lisible et compréhensible, disponible à partir de PHP 7.3.0 :

```php
setcookie('panier_id_produit', "425", [
    'expires' => time()+7200,
    'secure' => true,
    'httponly' => true
    ]
);
```

Nous utiliserons cette syntaxe pour les autres cookies, car il en faut un cookie par valeur, et nous avons quatre valeurs dans le panier, un identifiant de produit, un nom de produit, un prix et une quantité.

```php
setcookie('panier_produit_425_nom', "teeshirt", [
    'expires' => time() + 7200,
    'secure' => true,
    'httponly' => true
]);
setcookie('panier_produit_425_prix', "25", [
    'expires' => time() + 7200,
    'secure' => true,
    'httponly' => true
]);
setcookie('panier_produit_425_quantite', "2", [
    'expires' => time() + 7200,
    'secure' => true,
    'httponly' => true
]);
```

Pour supprimer un cookie, vous devrez en fait utiliser la même fonction ***setcookie()*** et inverser la date d'expiration :

```php
setcookie('panier_id_produit', "425", [
    'expires' => time()-7200
]);
```

Cela aura pour effet de supprimer ce cookie précisément, et pas un autre. Il faudra refaire le même code pour un autre cookie, avec son nom.

Il faut bien noter que la variable **$_COOKIE** stocke la liste des cookies renvoyés par le navigateur. Lorsqu'un utilisateur demande à accéder à notre page pour la première fois, le cookie *panier_id_produit* est créé côté serveur et est renvoyé au navigateur afin qu'il soit stocké sur la machine du visiteur.

Ainsi, la première fois qu'un utilisateur demande notre page, la variable **$_COOKIE** ne stocke pas encore notre cookie puisque celui-ci n'a pas encore été créé et donc le navigateur du visiteur ne peut rien renvoyer. Lors du premier affichage de la page nous avons donc des warnings car les valeurs sont inexistantes encore.

Si on actualise ensuite la page, en revanche, le navigateur renvoie bien cette fois-ci la valeur de notre cookie et son nom et celui-ci est bien stocké dans **$_COOKIE**. Notre page Web se recharge et on voit à présent les valeurs des différents cookies.

Travaux pratiques

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

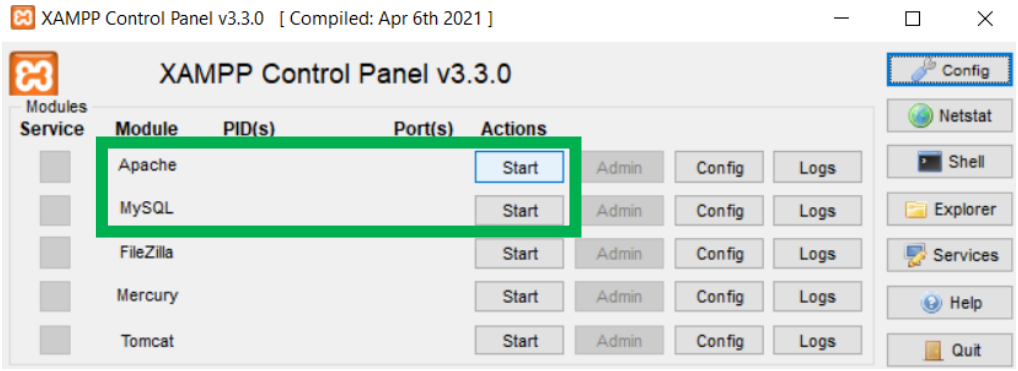
Utilisation d'une base de données MySQL

SOMMAIRE :

- Bases MySQL.
- Tables MySQL.
- Champs MySQL.
- Enregistrements MySQL.
- Fonctions PHP MySQL.
- Introduction au langage SQL (sélection, modification, suppression).
- Traitement des résultats des requêtes.

Nous allons créer notre première base de données, grâce au logiciel Xampp (ou Wamp/Mamp/Lamp) : https://www.apachefriends.org/fr/index.html

Pour cela, allumer le serveur local en cliquant sur le bouton "Start" pour Apache et MySQL.



Ensuite nous irons dans le navigateur, à l'adresse local : http://localhost/phpmyadmin/

Cette interface nous permet de gérer et manipuler toutes nos BDD ! Très pratique pour créer rapidement une base de données.

C'est ce que nous allons, en cliquant sur le bouton en haut, dans la barre de *nav* 

On lui donnera le nom de "entreprise". Une fois la BDD créée, nou sallons créer une première table, en paramétrant le nom des colonnes et le type de données que chaque colonne pourra contenir.

Quand cela est fait, on pourra insérer nos premières données, qui vous seront fournis dans un fichier texte.

Ensuite nous irons créer un nouveau fichier ***connexion_bdd.php*** dans le dossier **TP5**. Dans ce fichier php, nous ferons notre connexion entre le projet et la base de données, à l'aide de la classe PDO fourni par PHP :

```php
$connexion = new PDO( dsn: 'mysql:host=localhost;dbname=entreprise', username: 'root', password: '');
```

Les arguments attendus sont :

- Le nom du serveur = localhost
- Le nom de la base de données = entreprise
- Le nom de l'utilisateur = root (par défaut)
- Le mot de passe = vide par défaut (sous wamp, le mot de passe est vide - sous mamp, il sera nécessaire d'écrire "root")

À partir de maintenant, nous pourrons lire et manipuler les données contenues dans cette BDD entreprise à l'aide de PHP.

Nous allons d'abord voir les actions possibles sur cette table 'employes' et ses données : La lecture, la création, la modification et la suppression :

```php
// Création/Insertion
$requete_insertion = $connexion->query( statement: "INSERT INTO employes VALUES
(NULL, 'Jonathan', 'Gredler', 'm', 'informatique', '2022-11-16', 2000) ");
$resultat_insert = $requete_insertion->execute();
var_dump($resultat_insert);

// Modification
$requete_modification = $connexion->query( statement: "UPDATE employes SET salaire=2300 WHERE id_employes=991 ;");
$resultat_modif = $requete_modification->execute();
var_dump($resultat_modif);

// Suppression
$requete_suppression = $connexion->query( statement: "DELETE FROM employes WHERE id_employes=991;");
$resultat_supp = $requete_suppression->execute();
var_dump($resultat_supp);
```

La lecture est un peu différente dans sa syntaxe :

```php
// Lecture/Affichage
$requete_affichage = $connexion->query( statement: "SELECT * FROM employes;");
$resultats_affich = $requete_affichage->fetchAll( mode: PDO::FETCH_ASSOC);
var_dump($resultats_affich);
```

Maintenant que nous avons récupéré toutes les lignes de données et les avons stockés dans une variable ***$resultats_affich,*** affichons toutes les valeurs contenues sous forme de carte à l'aide de HTML et d'une boucle PHP !

```php
<?php if(!empty($resultats_affich) && isset($resultats_affich)) : ?>
    <?php foreach($resultats_affich as $employe) : ?>

        <div class="col-3 my-2">
            <div class="card" style="width: 18rem;">
                <div class="card-body">
                    <h5 class="card-title">Fiche employé n°<?= $employe['id_employes'] ?></h5>
                    <p class="card-text"><?= $employe['prenom'] . ' ' . $employe['nom'] ?></p>
                </div>
                <ul class="list-group list-group-flush">
                    <li class="list-group-item"><?= $employe['civilite'] === 'm' ? 'homme' : 'femme' ?></li>
                    <li class="list-group-item">Service : <?= $employe['service'] ?></li>
                    <li class="list-group-item">Embauché le : <?= $employe['date_embauche'] ?></li>
                    <li class="list-group-item">Salaire : <?= $employe['salaire'] ?></li>
                </ul>
                <div class="card-body text-center">
                    <a href="connexion_bdd.php?action=supprimer&id=<?= $employe['id_employes'] ?>"
                       class="text-danger btn btn-sm"
                       onclick="return confirm('Voulez-vous supprimer définitivement cet employé ?')">
                        Supprimer l'employé <i class="bi bi-x-circle-fill"></i>
                    </a>
                </div>
            </div>
        </div>

    <?php endforeach; ?>
<?php endif; ?>
```

Ce bout de code HTML et PHP sera répété autant de fois qu'il y a de résultats grâce à la boucle ***foreach()***.
Cela créera des fiches à la volée, de manière dynamique ! Et nous pouvons même supprimer une fiche 😉

Travaux pratiques

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

SOMMAIRE :

- Moteur de recherche : formulaire en relation avec une base de données.
- Fonctions avancées de sélection : recherches et tris.

Tout d'abord nous allons créer un nouveau fichier ***moteur-recherche.php*** dans le dossier **TP6**.

Déclarons un doctype HTML avec le CDN de Bootstrap 5 : https://getbootstrap.com/docs/5.0/getting-started/introduction/ .

Ajoutons ce code maintenant pour avoir un formulaire de recherche :

```
<h1 class="text-center my-4">Bienvenue dans le moteur de recherche</h1>

<div class="row">
    <div class="col-6 mx-auto mb-5">
        <form method="get" action="accueil.php" class="d-flex">
            <input class="form-control me-2"
                   type="search"
                   name="search_query"
                   placeholder="Rechercher par prénom, nom ou service"
                   aria-label="Search">
            <button class="btn btn-outline-success" type="submit">Rechercher</button>
        </form>
    </div>
</div>
```

Notez la méthode 'GET' du formulaire HTML. Cela permet de retrouver la valeur de l'input sous le nom écrit dans l'attribut *'name'* de la balise <input> dans la superglobale *$_GET* !

Maintenant que le HTML est prêt, ajoutons la ligne de code PHP pour créer une connexion à la BDD avec PDO.

Juste dessous, ajoutons une condition et un dump pour vérifier que l'on passe bien dans le ***if*** :

```
$connexion = new PDO( dsn: 'mysql:host=localhost;dbname=entreprise', username: 'root', password: '');

if(isset($_GET['search_query']) && !empty($_GET['search_query'])) {
    var_dump($_GET);
}
```

Maintenant que nous avons la valeur saisie pour la recherche dans $_GET, faisons un extract() de la superglobale pour avoir une variable :

```php
extract( &array: $_GET);
$requete = $connexion->query(
        statement: "SELECT * FROM employes WHERE
        prenom LIKE '%$search_query%' OR
        nom LIKE '%$search_query%' OR
        service LIKE '%$search_query%' ;"
);
```

Cette requête tolère les différences entre minuscules et majuscules, et nous permet de rechercher sur trois colonnes en même temps dans la BDD.

On va exécuter cette requête maintenant, dans une condition ***if*** de manière à gérer les erreurs :

```php
if($requete->execute() !== false) {
    $resultats_employes = $requete->fetchAll( mode: PDO::FETCH_ASSOC);
}
```

Ajoutons maintenant la partie HTML pour afficher les résultats stockés dans la variable ***$resultats_employes*** :

```php
<?php if(isset($resultats_employes) && !empty($resultats_employes)) : ?>

    <div class="d-flex justify-content-center ms-3">

        <?php foreach($resultats_employes as $employe) : ?>

            <div class="col mb-4">
                <div class="card" style="width: 15rem;">
                    <div class="card-body">
                        <h5 class="card-title">Fiche employé n°<?= $employe['id_employes'] ?></h5>
                        <p class="card-text"><?= $employe['prenom'] . ' ' . $employe['nom'] ?>,
                            <?= $employe['civilite'] === 'm' ? 'homme' : 'femme' ?></p>
                    </div>
                    <ul class="list-group list-group-flush">
                        <li class="list-group-item">Service : <?= $employe['service'] ?></li>
                        <li class="list-group-item">Embauché le : <?= $employe['date_embauche'] ?></li>
                        <li class="list-group-item">Salaire : <?= $employe['salaire'] ?></li>
                    </ul>
                </div>
            </div>
        <?php endforeach; ?>
```

Pour effectuer un tri sur les résultats de la recherche, ajoutons une clause ORDER BY dans la requête SQL ***$requete*** pour créer un tri, dans décroissant :

```
ORDER BY date_embauche DESC;"
```

Travaux pratiques

Réalisation d'un moteur de recherche : la sélection sur prénom, nom et service donne une liste de fiches correspondantes.
Implémentation multicouche.

Le graphisme en PHP

SOMMAIRE :

- Présentation de la librairie GD2.
- Création d'image, réutilisation.
- Gestion des polices et de l'écriture en mode image.
- Superposition de texte pour protection de droits.
- Intégration au site.
- Réalisation de graphiques statistiques.

Grâce à la bibliothèque GD2, PHP peut créer des images, fusionner des images, ajouter du texte sur une image et bien plus !

Regardons de plus près comment créer une image et l'utiliser dans une page web, en combinaison avec HTML. Créez un fichier ***images.html*** dans le dossier **TP7**.

Faisons un doctype HTML et un premier titre "Les images avec GD2 en PHP". En-dessous de ce titre mettons une balise <img> et dans l'attribut 'src' ajoutons ceci :

```
<h3>Créer une image</h3>
<img src="image.php" alt="">
```

Si vous trouvez étonnant de mettre en source un fichier php, vous allez comprendre avec la suite. C'est ce fichier image.php qui va créer l'image et la renvoyer à la balise <img>.

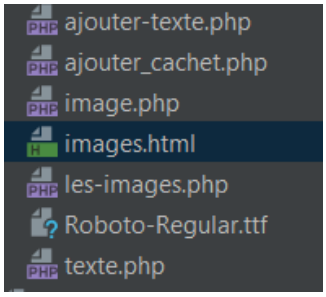
Regardons ce fichier PHP justement :

```php
<?php
// Définit la variable d'environnement pour GD
putenv( assignment: 'GDFONTPATH=' . realpath( path: '.'));

// Création d'une image de 300x100 pixels
$im = imagecreatetruecolor( width: 300, height: 300);
$red = imagecolorallocate($im, red: 0xFF, green: 0x00, blue: 0x00);
$black = imagecolorallocate($im, red: 0x00, green: 0x00, blue: 0x00);

// Définit l'arrière-plan en rouge
imagefilledrectangle($im, x1: 0, y1: 0, x2: 300, y2: 99, $red);

// Affichage de l'image sur le navigateur
header( header: 'Content-Type: image/png');

imagepng($im);
imagedestroy($im); /* Pour libérer la mémoire */
?>
```

Ce bout de code va générer une image avec un bloc de couleur noire en haut et le reste en rouge, d'une taille de 300 x 300 pixels :

Créer une image

Découvrons comment ajouter du texte dans cette image, avec une police de notre choix, et dont le fichier est dans notre projet '*Roboto-Regular.ttf*' :



Ajoutons une nouvelle balise <img> dans notre fichier HTML :

```html
<h3>Créer une image avec du texte</h3>
<img src="texte.php" alt="">
```

Créons le nouveau fichier PHP que l'on voit dans la source de la balise et ajoutons ce code :

```php
<?php
// Définit la variable d'environnement pour GD
putenv( assignment: 'GDFONTPATH=' . realpath( path: '.'));

// Chemin vers notre fichier de police ttf
$fontFile = __DIR__ . '/Roboto-Regular.ttf';

// Création d'une image de 300x100 pixels
$im = imagecreatetruecolor( width: 300, height: 300);
$red = imagecolorallocate($im, red: 0xFF, green: 0x00, blue: 0x00);
$black = imagecolorallocate($im, red: 0x00, green: 0x00, blue: 0x00);

// Définit l'arrière-plan en rouge
imagefilledrectangle($im, x1: 0, y1: 0, x2: 300, y2: 99, $red);


// Dessine le texte 'PHP Manual' en utilisant une police de taille 13
imagettftext($im, size: 13, angle: 0, x: 10, y: 55, $black, $fontFile, text: 'Image générée avec GD - Copyright');

// Affichage de l'image sur le navigateur
header( header: 'Content-Type: image/png');

imagepng($im);
imagedestroy($im); /* Pour libérer la mémoire */
?>
```

C'est sensiblement le même code, avec quelques lignes en plus :

- Les deux premières sont pour la police de caractères que nous utiliserons dans cet exemple,
- La fonction *imagettftext()* créée une image avec le texte en dernier argument.

Le résultat obtenu nous donne ce que l'on souhaite :

Abordons maintenant la création d'une fusion de 2 images, comme un logo à appliquer sur une image.
Ajoutons un nouveau titre et une balise <img> :

```
<h3>Ajouter un cachet à une photo</h3>
<img src="ajouter_cachet.php" alt="" width="680" height="440">
```

Il y a une largeur et hauteur définies en dur dans la balise, pour éviter les temps de chargement trop long pour les grandes résolutions.

Maintenant créons le fichier à la racine du dossier TP7, nommez-le comme dans la source de la balise et ajoutons ce code :

```php
header( header: "Content-type: image/png"); //on envoie les infos au navigateur

//$source = imagecreatefrompng("uploads/gd-logo.png"); //on ouvre l'image source
$source = imagecreatefrompng( filename: "uploads/orsysNew.png"); //on ouvre l'image source
$destination = imagecreatefrompng( filename: "uploads/pont.png");

// Récupération des valeurs de la taille du fichier source
$details_src = getimagesize( filename: "uploads/orsysNew.png"); //on récupère les dimensions de l'image source

/* On utilise ceci pour calculer l'endroit où on va commencer à copier.
On choisit en bas de l'image : calcul de la différence de la hauteur de l'image de destination et de l'image source. */
$y = imagesy($destination)-imagesy($source);

# Appel de la fonction pour copier 2 images entre elles.
imagecopy($destination,$source, dst_x: 15, $y, src_x: 0, src_y: 0, $details_src[0], $details_src[1]);

imagepng($destination);
imagedestroy($destination);
imagedestroy($source);
?>
```

Observons le résultat sur notre page web :

Le logo a bien été incrusté à la photo !

Pour ajouter du texte à un fichier image, cela se passe presque pareil !

Ajoutons une nouvelle balise <img> à notre fichier HTML et créons le fichier PHP dont le nom est dans l'attribut src de la balise :

```html
<h3>Ajouter un texte simple à une photo</h3>
<img src="ajouter-texte.php" alt="" width="680" height="440">
```

Dans le fichier PHP nous allons écrire ce code, en notant que nous ne pourrons pas gérer la taille de la police :

```php
$nom_image = "uploads/pont.png";  // le nom de votre image avec l'extension jpeg
$texte = "Copyright";  // Le texte que vous désirez écrire sur l'image

header ( header: "Content-type: image/png");

// Création de l'image au format .png
$image = imagecreatefrompng($nom_image);

// Déclaration de la couleur de texte (noir ici)
$textColor = imagecolorallocate($image,  red: 255,  green: 255,  blue: 255);

// Variable contenant la valeur de l'opération pour le positionnement sur l'axe Y
$y = imagesy($image)-(48/2);

imagestring($image, font: 5, x: 15, $y, $texte, $textColor);
imagepng($image); # Affichage de l'image sur le navigateur, si aucun chemin renseigné en 2nd paramètre.
imagedestroy($image); /* Pour libérer la mémoire */
```

Voici le résultat sur le navigateur :

C’est une solution rapide mais peu esthétique et complétement illisible ! Regardons une nouvelle façon plutôt, qui nous permet de gérer la taille de la police, la couleur et sa transparence. Ajoutez cette balise <img> au fichier HTML :

```html
<h3>Ajouter un texte avec police à une photo</h3>
<img src="ajouter-texte-police.php" alt="" width="680" height="440">
```

Comme pour les fois précédentes, créez le fichier source à la racine du dossier TP7 et ajoutons ce code PHP :

```php
// Définit la variable d'environnement pour GD
putenv( assignment: 'GDFONTPATH=' . realpath( path: '.'));

// Chemin vers notre fichier de police ttf
$fontFile = __DIR__ . '/Roboto-Regular.ttf';

// Création de l'image au format .png
$image = imagecreatefrompng( filename: "uploads/pont.png");

// Définition de la couleur blanche pour le texte
$blanc = imagecolorallocatealpha($image,  red: 255, green: 255, blue: 255,  alpha: 70);

// Dessine le texte 'PHP Manual' en utilisant une police de taille 13
imagettftext($image, size: 32, angle: 0, x: 50,  y: imagesy($image)-50, $blanc, $fontFile, text: '© Copyright' . ' - ' . date( format: 'Y'));

// Affichage de l'image sur le navigateur
header( header: 'Content-Type: image/png');

imagepng($image);
imagedestroy($image); /* Pour libérer la mémoire */
```

Voici l'image générée sur le navigateur :

Voici un lien pour la création de graphiques statistiques en PHP avec GD2 :
https://andry.developpez.com/tutoriels/php/creation-graphes-statistiques-et-geometriques/?page=page_1

Travaux pratiques

Intégration des différents modules réalisés.
Affichage d'image avec mention de Copyright.