Cours Framework de Développement Web

Nahla Haddar & Amal Bouaziz
Université de Sfax
Membre du laboratoire MIRACL

Audience: D-LSI-ADBD
Année-universitaire: 2023-2024

Avant-propos

- Dans ce cours vous allez apprendre à développer des scripts PHP native et PHP orienté objets, puis à une stade avancée, vous allez apprendre à développer des applications web en utilisant le Framework **Laravel**.
- Avec Laravel, vous apprendrez à:
  - intégrer des vues avec le moteur de gabarits **Blade**,
  - manipuler une base de données à l'aide de l'**ORM** (Object–relational mapping) **Eloquent**
  - interagir avec vos utilisateurs à l'aide de **formulaires parfaitement intégrés et validés**.

Plan du cours

- Initiation au PHP native et PHP orienté objets
- Introduction et installation du framework Laravel,
- Principe de fonctionnement selon le modèle MVC (Model-View-Controller)
- Modes de routage et paramètres de substitution
- Création de contrôleur (Controller)
- Création de templates (View) avec Blade ,
- Les formulaires: création et validation
- Manipulation de BD avec Eloquent ,
- Création d'application CRUD (Create, Read, Update, Delete)

Introduction

Sites Web

- Un site Web est un ensemble de pages Web stockés dans un serveur Web (tels que Apache).

- Il existe deux types de sites Web:
  - Les sites statiques
  - Les sites dynamiques

Les sites statiques

- Sont réalisés uniquement à l'aide des langages **HTML, CSS** et **JavaScript** (site vitrine),

- Leur contenu ne peut pas être changé que par l'intervention du webmaster.

Les sites dynamiques

- Utilisent HTML + CSS + JavaScript + des langages web dynamique (PHP, JSP/Servlet, Python…).
- Leur contenu est dit « dynamique » parce qu'il est stocké dans une **base de données** et il peut changer sans l'intervention du webmaster.
- Cette base de données doit être obligatoirement située dans un **serveur de base de données** (tels que MySql, SQLServer, etc.).

Client … Serveur

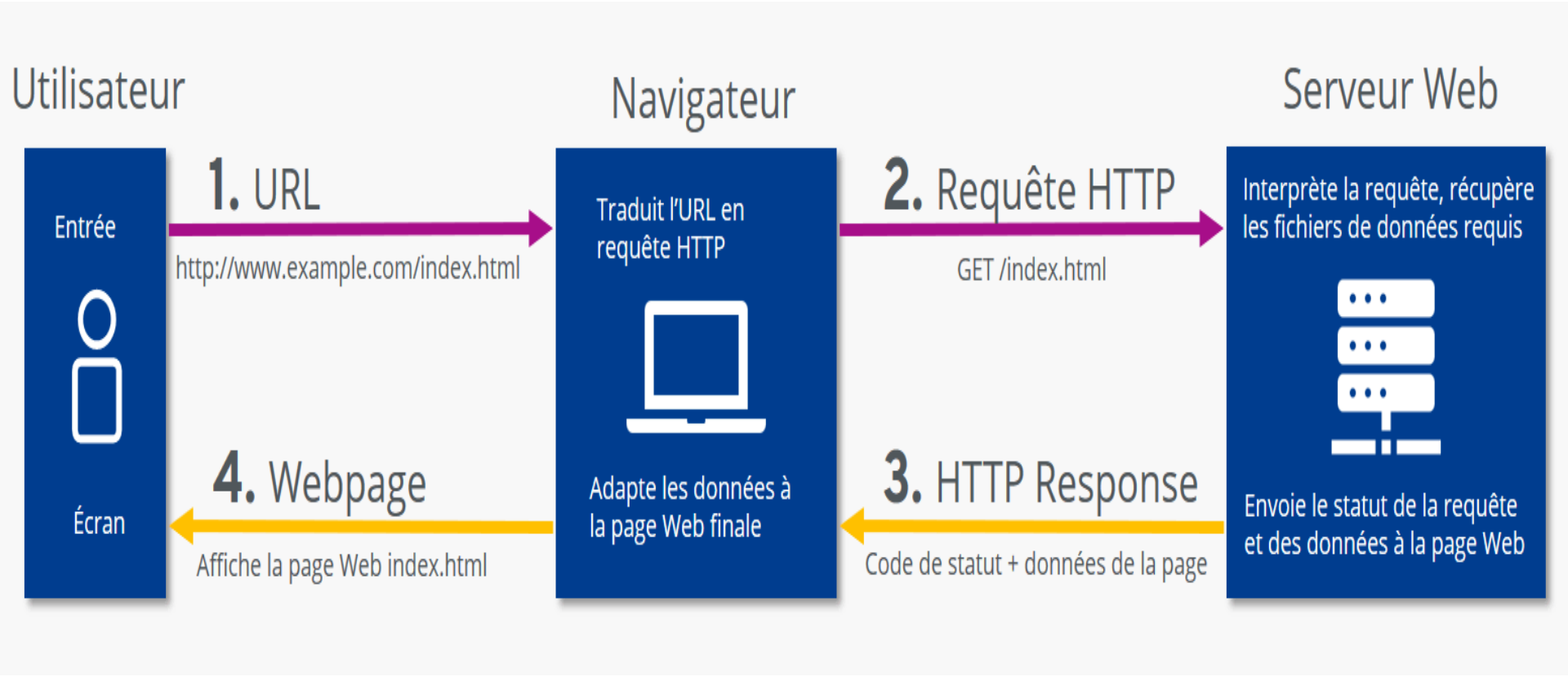
- Un **Serveur** est une application située sur un ordinateur très puissant, capable de gérer un grand nombre de requêtes simultanément.

- Un **Client** est une application qui se connecte à un serveur web pour obtenir ou modifier des informations à l'aide de requêtes (par exemple le navigateur web).

Le protocole HTTP

- HTTP (Hypertext Transfer Protocol) est un protocole de communication entre un client et un serveur.
- Le client demande une page (ressource) au serveur en envoyant une requête et le serveur réagit en envoyant une réponse, qui est en général une page Html.
- Quand on surfe sur Internet chacun de nos clics provoque en général cet échange.

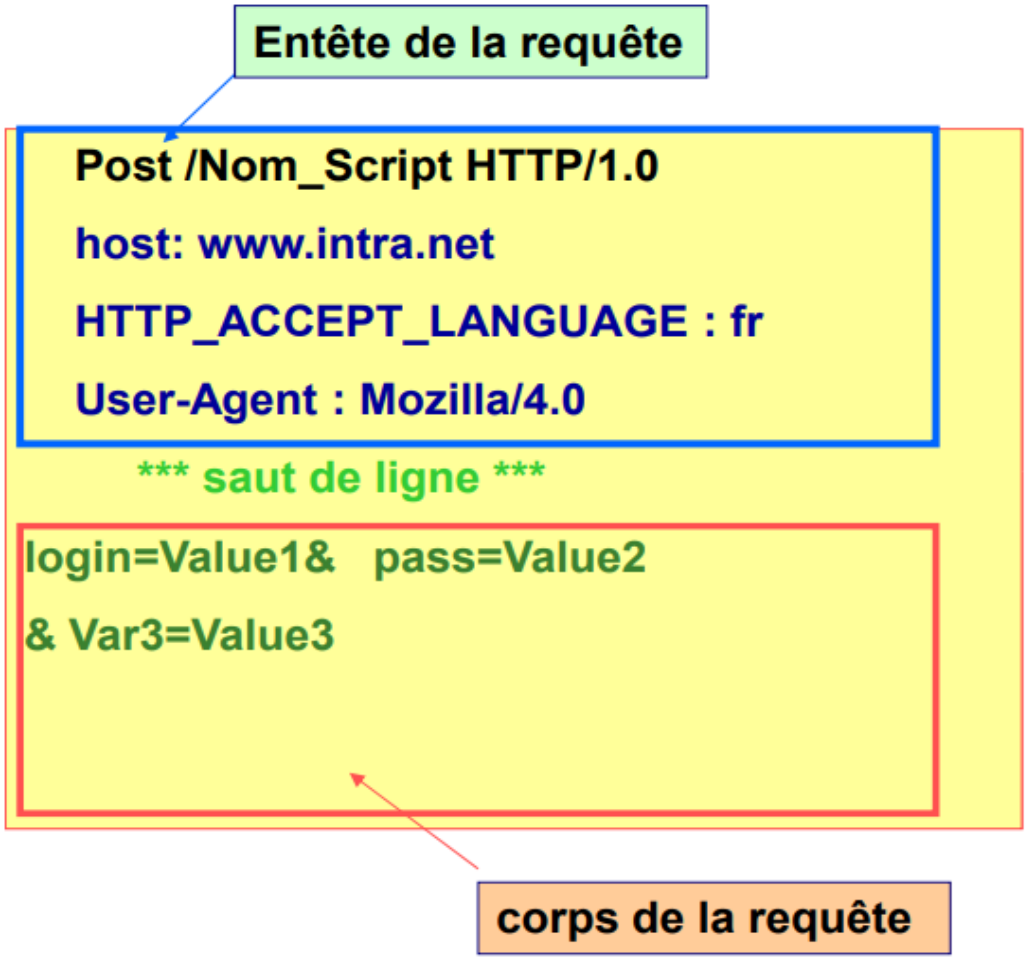
Processus de communication HTTP



Dialogue HTTP

- Deux types de dialogue:
  - **Récupération d'un document** (par le clic sur un lien ou l'écriture de son URL dans la barre d'adresse du navigateur)
    - méthode **GET**
  - **Soumission d'un formulaire**
    - méthodes **GET** ou **POST**

Requête HTTP de type POST



Méthode,chemin,version

Nom de domaine

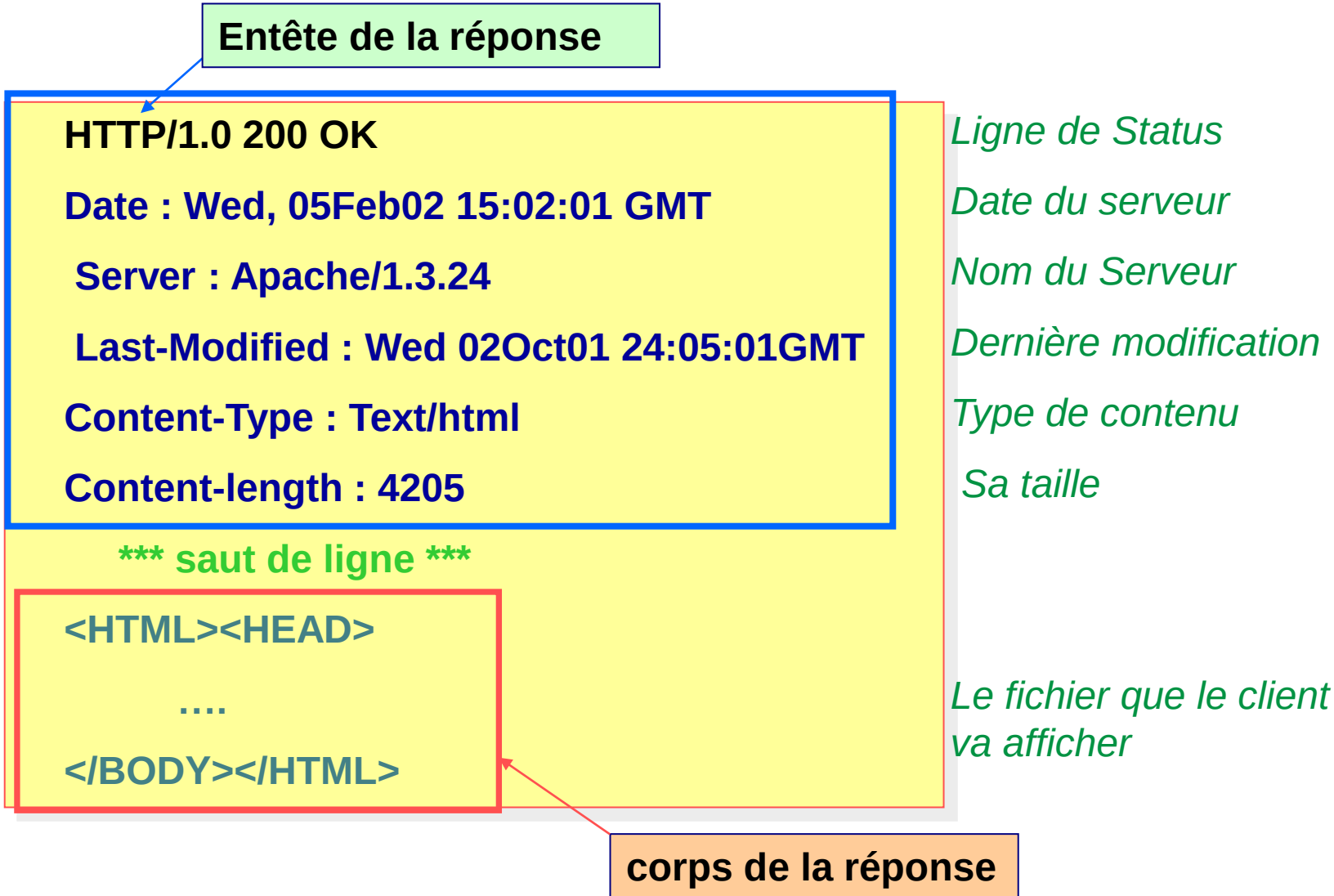
Code de la langue

Type et version du navigateur

Paramètres des différents champs du formulaire.

Requête HTTP de type GET

Réponse du serveur



Réponse du serveur

- De très nombreux statuts existent, parmi les plus connus :
  - 200, la page a été retournée sans erreur du serveur ;
  - 404, le code HTTP pour une ressource qui n'a pas été retrouvée sur le serveur ;
  - les codes 3XX, qui signalent les redirections de ressources ;
  - les codes 4XX, qui signalent une erreur côté utilisateur/client ;
  - les codes 5XX, qui signalent une erreur côté serveur.

PHP : Les bases du langage

Caractéristiques Principales

- **PHP:** Signifie d'abord Personal Home Pages puis HypertextPreProcessor
- **Langage interprété**
  - Son interpréteur se nomme **Zend Engine**
  - Pas de compilation
  - Exécuté instruction par instruction
  - Multi-plateformes
- Spécialisé dans la génération de texte ou de documents
  - HTML, PDF, Images
- **Fichiers d'extension .php**
  - Code PHP+ balises HTML

Imbrication de code HTML et PHP

- Une page PHP peut être entièrement programmée en PHP ou mélangée avec du code html.

- Pour différencier le code PHP des balises HTML, on utilise des balises particulières:
  - `<?php … ?>` (forme préférée)
  - `<script language="php">… </script>`

Exemple simple

Script.php

```
<!DOCTYPE html>
<html>
<body>

<?php
echo "My first PHP script!";
?>

</body>
</html>
```

Résultat:

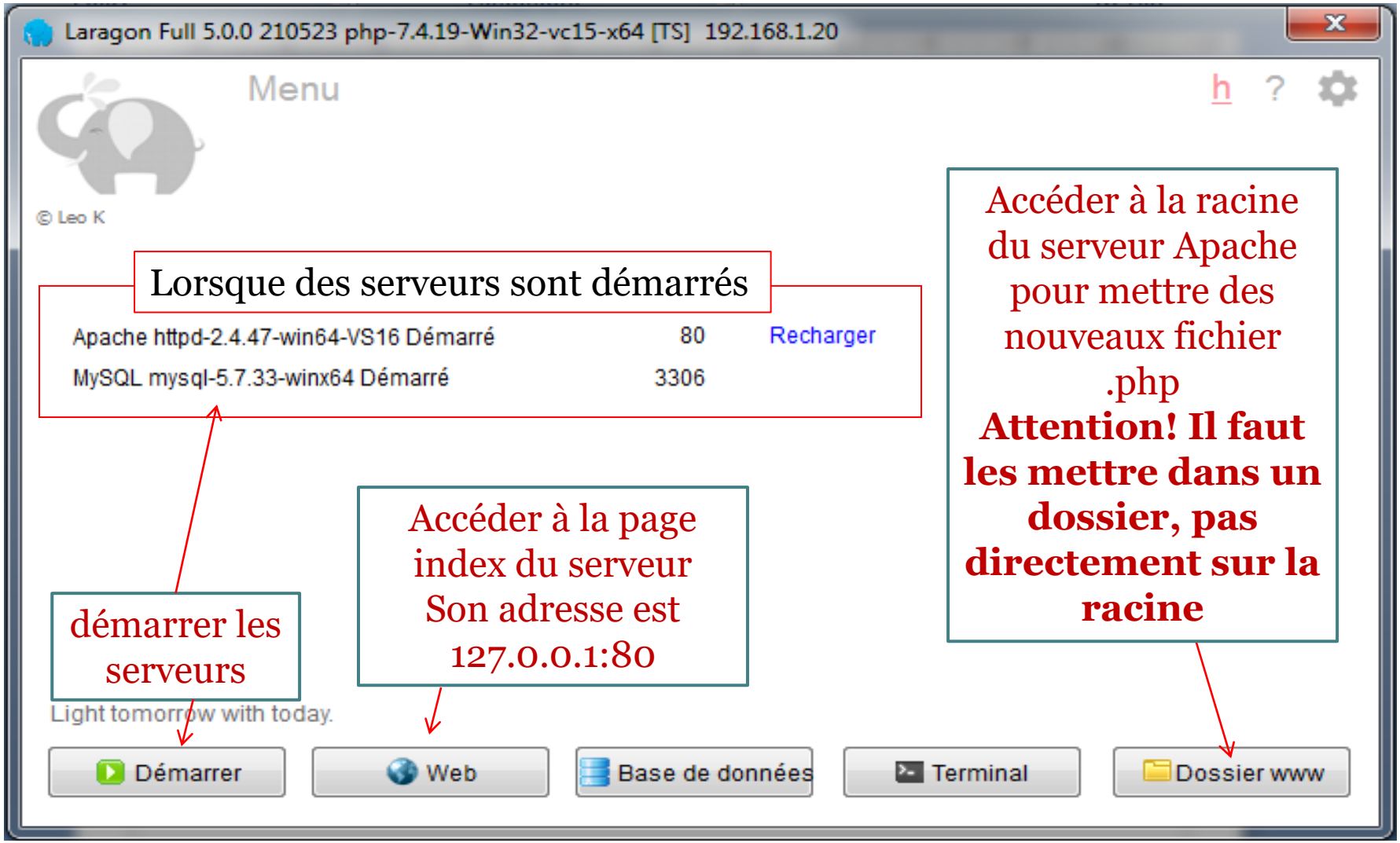
My first PHP script!

Comment tester notre code ?

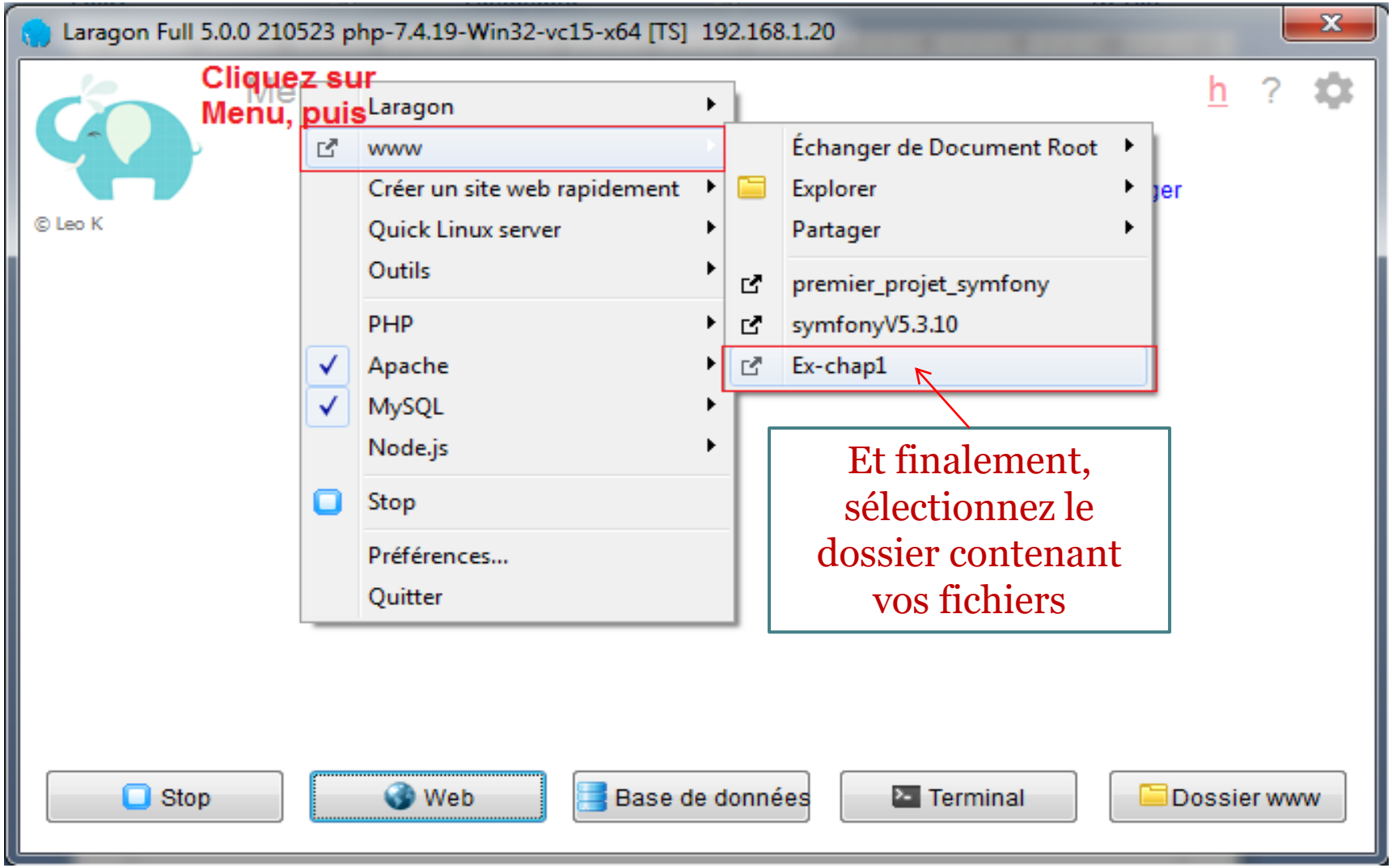
Installation de l'environnement de travail

- Deux solutions sont possibles:
  - **1ère solution:**
    - Installer un environnement web en local tel que **XAMPP, EASYPHP, CADDY,...** Cela vous permettra d'avoir **PHP, Apache** et **MySQL** installés en local
  - **2ème solution (préférable):**
    - Installer **Laragon**: https://laragon.org/download/index.html un environnement de développement web assez complet:
      - Offre un **serveur Apache, serveur de BD (MySQL, PostgresSQL, NoSQL, des logiciels de gestions de BD (PhpMyAdmin, RoboMongo, PgAdmin), Composer**, ... et encore beaucoup plus.

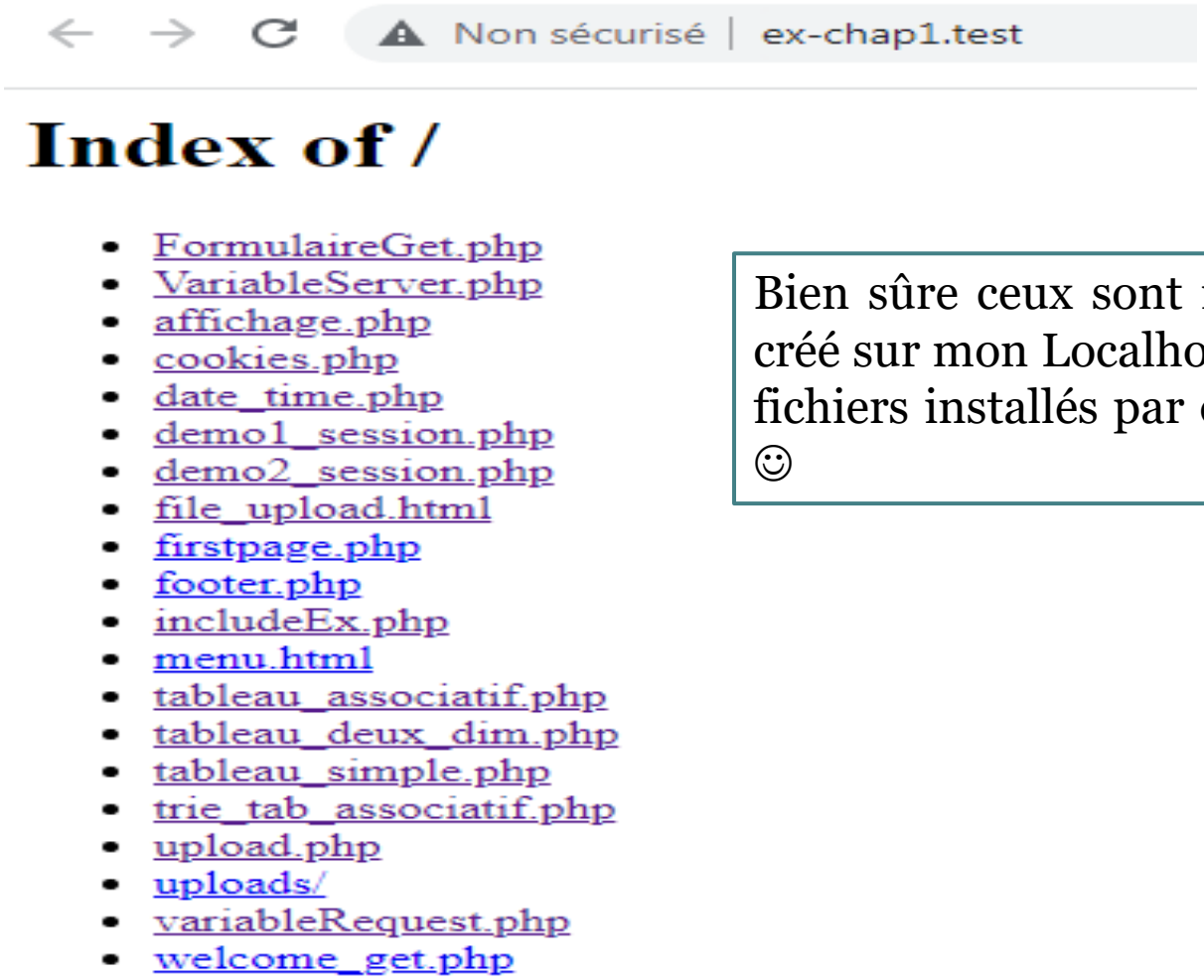
Prise en main de Laragon



Comment tester vos fichier.php ?



Cliquez sur l'un des fichiers pour voir le résultat



Bien sûre ceux sont mes fichiers que j'ai créé sur mon Localhost, ceux sont pas des fichiers installés par défaut dans Laragon ☺

Les commentaires

- Les commentaires s'utilisent comme en C et en C++ avec :
  - /* commentaire sur plusieurs ligne*/
  - et // ou # commentaire sur une seule ligne
- **Exemple:**

```
<!DOCTYPE html>
<html>
<body>

<?php
// This is a single-line comment

# This is also a single-line comment
?>

</body>
</html>
```

Affichage sur écran

- `echo` et `print` sont plus ou moins les mêmes.
- Ils sont tous deux utilisés pour afficher des données à l'écran.
- `echo` n'a pas de valeur de retour alors que `print` a une valeur de retour de 1
- `echo` peut avoir plusieurs paramètres alors que `print` non
- `echo` est légèrement plus rapide que `print`

Affichage sur écran

- **Affichage de texte (peut contenir des balises HTML)**

```php
<?php
echo "<h2>PHP is Fun!</h2>";
echo "Hello world!<br>";
echo "I'm about to learn PHP!<br>";
echo "This ", "string ", "was ", "made ", "with multiple parameters.";
?>
```

PHP is Fun!

Hello world!
I'm about to learn PHP!
This string was made with multiple parameters.

```php
<?php
print "<h2>PHP is Fun!</h2>";
print "Hello world!<br>";
print "I'm about to learn PHP!";
?>
```

PHP is Fun!

Hello world!
I'm about to learn PHP!

Les variables en PHP:
1. Déclaration des variables

- Une variable commence par un dollar $ suivi d'un nom de variable.
- Les variables ne sont pas typées au moment de leur création.

```php
<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
?>
```

- Attention PHP est sensible à la casse : var et Var ne sont pas les mêmes variables !
- **Les règles à respecter :**
  - Une variable peut commencer par une lettre
  - Une variable peut commencer par un souligné (underscore) « _ »
  - Une variable ne doit pas commencer par un chiffre.

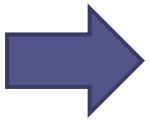
Les variables en PHP:
2. Affichage des variables

- Avec `echo` ou `print` même syntaxe:
  - Mais … attention au **guillemets** "…" ou **quottes** '…'

```php
<?php
$txt1 = "Learn PHP";
$txt2 = "ISIMS";
$x = 5;
$y = 4;

echo "<h2> $txt1 </h2>";
echo 'Study PHP at ' . $txt2 . '<br>';
echo "$x + $y = ". ($x + $y);
?>
```

Avec **les quottes** la **concaténation** entre les variables et le texte **est toujours obligatoires.**



Learn PHP

Study PHP at ISIMS
5 + 4 = 9

Avec **les guillemets** la **concaténation n'est obligatoire** que en cas d'évaluation d'une expression.

Les variables en PHP:

3. Les constantes

- Les constantes sont automatiquement globales et peuvent être utilisées dans l'ensemble du script.
- **Syntaxe:**
  - `define("nom_constante", valeur_constante )`
- **Exemples**:

Une chaine de caractère

```php
<?php
define("GREETING", "Welcome to W3Schools.com!");
?>
```

Un tableau de chaines de caractères

```php
<?php
define("cars", [
  "Alfa Romeo",
  "BMW",
  "Toyota"
]);
?>
```

Les variables en PHP:
4. Portée des variables

- PHP a trois portées de variables différentes :
  - Globale
  - Locale
  - Statique

Les variables en PHP:
4. Portée des variables (GLOBALE)

- Une variable déclarée en dehors d'une fonction **a une portée GLOBALE** et **n'est accessible qu'en dehors d'une fonction** :

```php
<?php
$x = 5; // global scope

function myTest() {
  // using x inside this function will generate an error
  echo "<p>Variable x inside function is: $x</p>";
}
myTest();

echo "<p>Variable x outside function is: $x</p>";
?>
```

Mais, normalement une variable globale est accessible dans les fonctions !
Comment doit-on faire alors ?

Les variables en PHP:
4. Portée des variables (GLOBALE)

- PHP stocke également toutes les variables globales dans un tableau appelé $GLOBALS[‘nom_var’].
- Ce tableau est également accessible depuis les fonctions et peut être utilisé pour mettre à jour directement les variables globales.

```php
<?php
$x = 5;
$y = 10;

function myTest() {
  $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
}

myTest();
echo $y; // outputs 15
?>
```

Les variables en PHP:
4. Portée des variables (LOCALE)

- Une variable déclarée dans une fonction a une PORTÉE LOCALE et n'est accessible qu'au sein de cette fonction :

```php
<?php
function myTest() {
  $x = 5; // local scope
  echo "<p>Variable x inside function is: $x</p>";
}
myTest();

// using x outside the function will generate an error
echo "<p>Variable x outside function is: $x</p>";
?>
```

Les variables en PHP:
4. Portée des variables (STATIQUE)

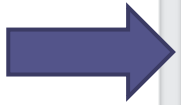
- Lorsqu'une **fonction est terminée/exécutée**, toutes **ses variables sont supprimées**.
- Cependant, nous voulons parfois qu'une variable locale **ne soit PAS supprimée**.
- Pour cela, utilisez le mot-clé **static** lors de la première déclaration de la variable :

```
<!DOCTYPE html>
<html>
<body>

<?php
function myTest() {
  static $x = 0;
  echo $x;
  $x++;
}

myTest();
echo "<br>";
myTest();
echo "<br>";
myTest();
?>

</body>
</html>
```



```
0
1
2
```

Les variables en PHP:
5. Fonctions sur des variables

- Vérifier l'existence d'une variable (isset)

```php
<?php
$a = "une variable en PHP";
if(isset($a)) echo "la variable a existe";
unset($a);
echo "la variable a a été supprimée ...";
```

- Tester si une variable est vide (empty)

```php
<?php
$a = "une variable en PHP";
if (!empty($a)) echo " La variable existe et elle n\'est␣
 ↪pas vide !";
```

  - Attention! La fonction empty() répond vrai si la variable n'existe pas et ceci sans faire aucun warning !

Les chaînes en PHP:
1. Les bases

- La chaine est traitée comme un tableau de caractères indexé par un entier => `$str="Hello" ;  echo $str[1]; \\affiche ‘e’`
- **La concaténation à l’aide de .**

```php
$str="Salut les Amis !\n";
$str.="Comment ça va ?"; // "Salut les Amis !\nComment ça va ?
$str2=$str."\n"; // "Salut les Amis !\nComment ça va ?\n
```

- **La longueur d’une chaine :**

```php
<?php
echo strlen("Hello world!"); // outputs 12
?>
```

- **Compter le nombre de mots d’une chaine**

```php
<?php
echo str_word_count("Hello world!"); // outputs 2
?>
```

Les chaînes en PHP:
2. Les fonctions

- **Mettre en majuscules/minuscules :**
  - avec **strtoupper()** pour obtenir des majuscules
  - avec **strtolower()** pour mettre en minuscules
  - avec **ucfirst()** pour mettre en majuscule la première lettre d'une chaine
  - avec **ucwords()** pour mettre en majuscule la première lettre de chaque mot dans une chaine
- **Exemple:**

```php
<?php
$str = "Marie A un Petit Agneau, et l'aime TRès fORt.";
$str = strtolower($str);
echo $str; // marie a un petit agneau, et l'aime très fort.
?>
```

Les chaînes en PHP:
2. Les fonctions

- **Remplace le texte dans une chaîne:**

```php
<?php
echo str_replace("world", "Dolly", "Hello world!"); // outputs Hello Dolly!
?>
```

- **Inverser une chaîne:**

```php
<?php
echo strrev("Hello world!"); // outputs !dlrow olleH
?>
```

- **Vérifier si une chaine est numérique:**

```php
$x ="5985";
echo is_numeric($x); // affiche True
```

Les chaînes en PHP:
3. Recherche de sous-chaines

- **stristr()**
  - Il recherche la première occurrence d'une chaîne à l'intérieur d'une autre chaîne et affiche la portion de cette dernière à partir de la première occurrence rencontrée.
  - Si la chaine n'existe pas, elle stristr() **retourne faux**
  - Cette fonction est insensible à la casse.

```php
<?php
  $email = 'USER@EXAMPLE.com';
  echo stristr($email, 'e'); // Affiche ER@EXAMPLE.com
  echo stristr($email, 'e', true); // Depuis PHP 5.3.0, Affiche US
?>
```

- **strstr()** fait le même travail mais elle est sensible à la casse.

Les chaînes en PHP:
Activité

- Transformez une chaîne écrite dans des casses différentes afin que chaque mot ait une initiale en majuscule.
- **Exemple:**

```
$ch="TransFOrmeZ unE ChaîNE écRITe dans des cASses diFFéreNTes afiN
qUe chAQue MOT ait une inITiale en MAJUSCULE";
```

- **Résultat:**

```
Transformez Une Chaîne Écrite Dans Des Casses Différentes Afin Que
Chaque Mot Ait Une Initiale En Majuscule
```

- **Solution:**<?php echo ucwords(strtolower($ch)); ?>

Date & Time

1. Date

- *La fonction **date()** spécifie comment formater la date (ou l'heure).*
- ***Syntaxe:***

```
date(string $format, ?int $timestamp = null): string
```

- **Paramètres les plus utilisés pour le format:**
  - **d** - Représente le jour du mois (01 à 31)
  - **m** - Représente un mois (01 à 12)
  - **Y** - Représente une année (en quatre chiffres)
  - **l** (minuscule 'L') - Représente le jour de la semaine
- D'autres caractères, comme "/", "." ou "-" peuvent également être insérés entre les caractères pour ajouter une mise en forme supplémentaire.
- Pour une liste complète de fonctions sur les dates: https://www.w3schools.com/php/php_ref_date.asp

Date & Time

1. Date (suite)

Today is 2022/01/24
Today is 2022.01.24
Today is 2022-01-24
Today is Monday

- ***Exemple 1:***

```php
<?php
echo "Today is " . date("Y/m/d") . "<br>";
echo "Today is " . date("Y.m.d") . "<br>";
echo "Today is " . date("Y-m-d") . "<br>";
echo "Today is " . date("l");
?>
```

- **Exemple 2:** CopyRight automatique

```
&copy; 2010-<?php echo date("Y");?>
```

© 2010-2022

Date & Time
2. Time

- **Paramètres (les plus utilisés) de la fonction date pour obtenir le temps:**
  - **H** - Format 24 heures d'une heure (00 à 23)
  - **h** - format 12 heures d'une heure avec des zéros non significatifs (01 à 12)
  - **i** - Minutes avec des zéros non significatifs (00 à 59)
  - **s** - Secondes avec des zéros non significatifs (00 à 59)
  - **a** - Minuscule Ante meridiem et Post meridiem (am ou pm)

Date & Time
2. Time

- **Exemple:** définir le fuseau horaire sur "America/New_York", puis affiche l'heure actuelle dans le format spécifié :

```php
<?php
date_default_timezone_set("America/New_York");
echo "The time is " . date("h:i:sa");
?>
```

- **Résultat:**

The time is 01:49:11pm

Date & Time
La fonction time

- La fonction **time()** renvoie le temps écoulé en nombre de secondes depuis l'époque Unix (1er janvier 1970 00:00:00 GMT) **jusqu'au moment actuel**

  => l'horodatage Unix.
- **Syntaxe:**
  - *time()*
- **Exemple:** renvoie l'heure actuelle sous la forme d'un horodatage Unix, puis formate-la en une date :

```php
<?php
    $t=time();
    echo($t . "<br/>");
    echo(date("Y-m-d h:i:sa",$t));
?>
```

1644329327
2022-02-08 02:08:47pm

Date & Time
La fonction mktime

- La fonction **mktime()** renvoie l'horodatage Unix **jusqu'à une date et une heure donnée**
- **Syntaxe:**
  - *mktime(hour, minute, second, month, day, year)*
- **Exemple:** trouver le jour d'une date donnée

```php
<?php
// Prints: October 3, 1975 was on a Friday
echo "Oct 3, 1975 was on a ".date("l", mktime(0,0,0,10,3,1975));
?>
```

Oct 3, 1975 was on a Friday

Activité

- Créez un simple script de "compte à rebours d'anniversaire", le script comptera le nombre de jours entre le jour actuel et l'anniversaire.
- **Solution ?**

```php
<?php
$target_days = mktime(0,0,0,6,9,2022);
$today = time();
$diff_days = ($target_days - $today);
$days = (int)($diff_days/86400);
print "Il vous reste  $days jours jusqu'au votre prochain anniversaire!"."\n";
?>
```

- **Exécution:**

Il vous reste 126 jours jusqu'au votre prochain anniversaire!

Les tableaux en PHP

1. Tableaux simples

- Un tableau stocke plusieurs valeurs dans une seule variable
- **Exemple:**

```php
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".<br/>";
print_r($cars);# affiche le contenu d'un tableau
?>
```

  - Les **indices** des cases sont des entiers allons de **0** à **count($cars)-1**
- **Résultat:**

```
I like Volvo, BMW and Toyota.
Array ( [0] => Volvo [1] => BMW [2] => Toyota )
```

Les tableaux en PHP

2. Tableaux associatifs

- Les **indices** des cases sont **des chaines de caractères => les indices sont des clés nommés**
- Il existe deux manières de créer un tableau associatif :

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
```

- Ou

```
$age['Peter'] = "35";
$age['Ben'] = "37";
$age['Joe'] = "43";
```

- **Exemple qui affiche:** Peter is 35 years old.

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>
```

Les tableaux en PHP

3. Tableaux multidimensionnels

- Un tableau multidimensionnel est un tableau qui va lui-même contenir d'autres tableaux en valeurs.
  - **Tableau à deux dimensions:** un tableau qui contient un ou plusieurs tableaux en valeurs,
  - **Tableau à trois dimensions:** un tableau qui contient un ou plusieurs tableaux en valeurs, qui contiennent eux-mêmes d'autres tableaux en valeurs
  - **etc.**
- **Les « sous » tableaux** vont pouvoir être des tableaux **simples** ou des tableaux **associatifs** ou **un mélange des deux**.

Les tableaux en PHP

3. Tableaux multidimensionnels

- **Exemple:** Tableau à deux dimensions
  - Soit le tableau suivant:

Name	Stock	Sold
Volvo	22	18
BMW	15	13
Saab	5	2
Land Rover	17	15

  - On peut le stocker dans le tableau PHP suivant:

```
$cars = array (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
);
```

Les tableaux en PHP

3. Tableaux multidimensionnels

- **Exemple:** Tableau à deux dimensions
  - Affichage des éléments:

```php
<?php
$cars = array (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
);
// affichage
echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2].".<br>";
echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2].".<br>";
echo $cars[2][0].": In stock: ".$cars[2][1].", sold: ".$cars[2][2].".<br>";
echo $cars[3][0].": In stock: ".$cars[3][1].", sold: ".$cars[3][2].".<br>";
?>
```

Exécution:

Volvo: In stock: 22, sold: 18.
BMW: In stock: 15, sold: 13.
Saab: In stock: 5, sold: 2.
Land Rover: In stock: 17, sold: 15.

Les tableaux en PHP:
4. Trie des tableaux

- `sort($tab)` - trie un tableau simple $tab par ordre croissant
- `rsort($tab)` - trie un tableau simple $tab par ordre décroissant
- `asort($tab)` - trie un tableau associatif $tab par ordre croissant, selon la valeur
- `ksort($tab)` - trie un tableau associatif $tab par ordre croissant, selon la clé
- `arsort()` - trie un tableau associatif $tab par ordre décroissant, selon la valeur
- `krsort()` - trie un tableau associatif $tab par ordre décroissant, selon la clé

Les tableaux en PHP:
4. Trie des tableaux

- Exemple:

```
<body>
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
arsort($age);
print_r($age) # Affiche des informations lisibles pour une variable
?>
</body>
```

Exécution:

Array ([Joe] => 43 [Ben] => 37 [Peter] => 35)

Les opérateurs en PHP

1. Opérateur arithmétique

Opérateur	Nom	Exemple
+	addition	`$x + $y`
-	Soustraction	`$x - $y`
*	Multiplication	`$x * $y`
/	Division	`$x / $y`
%	Module (reste de la division)	`$x % $y`
**	Exponentiation	`$x ** $y`

Les opérateurs en PHP

2. Opérateur d'affectation

Affectation	Équivalent à ...
`$x = $y`	`$x = $y`
`$x + = $y`	`$x= $x + $y`
`$x - = $y`	`$x= $x - $y`
`$x * = $y`	`$x= $x * $y`
`$x / = $y`	`$x= $x / $y`
`$x % = $y`	`$x= $x % $y`

Les opérateurs en PHP

3. Opérateur de comparaison

Opérateur	Exemple	Résultat
==	$x == $y	Renvoie true si $x est égal à $y
===	$x === $y	Renvoie true si $x est égal à $y et qu'ils sont du même type
!=	$x != $y	Renvoie true si $x n'est pas égal à $y
<>	$x <> $y	Renvoie true si $x n'est pas égal à $y
!==	$x !== $y	Renvoie true si $x n'est pas égal à $y, ou s'ils ne sont pas du même type
>	$x > $y	Renvoie true si $x est supérieur à $y
<	$x < $y	Renvoie true si $x est inférieur à $y
>=	$x >= $y	Renvoie true si $x est supérieur ou égal à $y
<=	$x <= $y	Renvoie true si $x est inférieur ou égal à $y

Les opérateurs en PHP

3. Opérateur logique

Opérateur	Exemple	Résultat
and	$x and $y	True si $x et $y sont vrais
or	$x or $y	True si $x ou $y est vrai, ou les deux
xor	$x xor $y	True si $x ou $y est vrai, mais pas les deux
&&	$x && $y	True si $x et $y sont vrais
||	$x || $y	True si $x ou $y est vrai
!	!$x	True si $x n'est pas vrai

Instructions conditionnelles

1. If...

- **Syntaxe:**

```
if (condition) {
  code to be executed if condition is true;
}
```

- **Exemple:**
  - **Sortie:** « Have a good day! » si l'heure actuelle ($t) est inférieure à 20 :

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

if ($t < "20") {
  echo "Have a good day!";
}
?>
```

Instructions conditionnelles
2. If... else

- **Syntaxe:**

```
if (condition) {
  code to be executed if condition is true;
} else {
  code to be executed if condition is false;
}
```

- **Exemple:**
  - **Sortie:** « Have a good day! » si l'heure actuelle ($t) est inférieure à 20, sinon, « Have a good night! »:

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

if ($t < "20") {
  echo "Have a good day!";
} else {
  echo "Have a good night!";
}
?>
```

Instructions conditionnelles

3. If...elseif

- **Syntaxe:**

```
if (condition) {
  code to be executed if this condition is true;
} elseif (condition) {
  code to be executed if first condition is false and this condition is true;
} else {
  code to be executed if all conditions are false;
}
```

- **Exemple:**

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

if ($t < "10") {
  echo "Have a good morning!";
} elseif ($t < "20") {
  echo "Have a good day!";
} else {
  echo "Have a good night!";
}
?>
```

Instructions conditionnelles

4. Switch

- Selon la valeur d'une variable, un seul bloc sera exécuté,
- Si aucune correspondance entre les blocs et la valeur, le bloc par défaut sera exécuté.

```php
<?php
    $note = 10;

    switch ($note){
        case 0:
            echo 'Vous avez obtenu la note de 0';
            break;
        case 5:
            echo 'Vous avez obtenu la note de 5';
            break;
        case 10:
            echo 'Vous avez obtenu la note de 10';
            break;
        case 15:
            echo 'Vous avez obtenu la note de 15';
            break;
        case 20:
            echo 'Vous avez obtenu la note de 20';
            break;
        default:
            echo 'Je n\'ai rien à afficher pour votre note';
    }
?>
```

Instructions conditionnelles

4. Switch (Activité)

- **Activité:** Créez une instruction switch qui affichera "Hello" si $color est "red" et "welcome" si $color est "green".
- **Solution ?**

```php
<?php
$colors=array("red", "green");
$indice=rand(0,1);
$color=$colors[$indice];
switch ($color){
    case "red":
        echo "Hello !";
        break;
    case "green":
        echo "Welcome!";
        break;
}
?>
```

Les boucles

- **while -** exécute un bloc de code tant que la condition spécifiée est vraie
- **do...while** – exécute un bloc de code une fois, puis répète la boucle tant que la condition spécifiée est vraie
- **for** – exécute un bloc de code un nombre de fois spécifié
- **foreach** – exécute un bloc de code pour chaque élément d'un tableau

Les boucles

1. while

- **Syntaxe:**

```
while (condition is true) {
  code to be executed;
}
```

- ***Exemple:***

```
<?php
$x = 1;

while($x <= 5) {
  echo "The number is: $x <br>";
  $x++;
}
?>
```

Exécution:

The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5

Les boucles
2. do...while

- **Syntaxe:**

```
do {
  code to be executed;
} while (condition is true);
```

- ***Exemple:***

```
<?php
$x = 1;

do {
  echo "The number is: $x <br>";
  $x+=10;
} while ($x <= 100);
?>
```

Exécution:

```
The number is: 1
The number is: 11
The number is: 21
The number is: 31
The number is: 41
The number is: 51
The number is: 61
The number is: 71
The number is: 81
The number is: 91
```

Les boucles

3. for

- **Syntaxe:**

```
for (init counter; test counter; increment counter) {
  code to be executed for each iteration;
}
```

- ***Exemple:***

```php
<?php
for ($x = 0; $x <= 100; $x+=10) {
  echo "The number is: $x <br>";
}
?>
```

Exécution:

The number is: 0
The number is: 10
The number is: 20
The number is: 30
The number is: 40
The number is: 50
The number is: 60
The number is: 70
The number is: 80
The number is: 90
The number is: 100

Les boucles
4. foreach

- **Syntaxe:**

```
foreach ($array as $value) {
   code to be executed;
}
```

- ***Exemple 1:***

```php
<?php
$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $value) {
  echo "$value <br>";
}
?>
```

Exécution:

red

green

blue

yellow

Les boucles
4. foreach (suite)

- ***Exemple 2:***

```php
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

foreach($age as $x => $val) {
  echo "$x = $val<br>";
}
?>
```

Exécution:

Peter = 35

Ben = 37

Joe = 43

Les boucles
Activité

- Soit le tableau à deux dimensions suivant:

```php
$cars = array (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
);
```

- Utilisez la boucle foreach pour afficher $cars comme suit:

Volvo: In stock: 22, sold: 18.
BMW: In stock: 15, sold: 13.
Saab: In stock: 5, sold: 2.
Land Rover: In stock: 17, sold: 15.

Les boucles
Activité

- **Solution ?**

```php
<?php
$cars = array (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
);
// parcourt
echo "<h3> affichage avec boucle </h3>";
foreach($cars as $car){
    echo $car[0].": In stock: ".$car[1].", sold: ".$car[2].".<br>";
}
```

Volvo: In stock: 22, sold: 18.
BMW: In stock: 15, sold: 13.
Saab: In stock: 5, sold: 2.
Land Rover: In stock: 17, sold: 15.

Inclusion de fichiers externes

- L'inclusion de fichiers est très utile lorsque vous souhaitez inclure le même PHP, HTML ou texte sur plusieurs pages d'un site Web.
- **Les fonctions utilisées:**
  - `require 'filename'` produira une erreur fatale (E_COMPILE_ERROR) et arrêtera le script si le fichier n'existe pas
  - `include 'filename'` ne produira qu'un avertissement (E_WARNING) et le script continuera si le fichier n'existe pas

Inclusion de fichiers externes

- **Exemple:**
  - Supposons que nous ayons un fichier de pied de page standard appelé « footer.php », qui ressemble à ceci :

```
<p> Copyright &copy; 2010-<?=date("Y")?>. isims.rnu.tn </p>
```

  - Pour inclure le fichier footer.php dans la page « home.php », utilisez l'instruction `include`:

```
<html>
<body>

<h1>Welcome to my home page!</h1>
<p>Some text.</p>
<p>Some more text.</p>
<?php include 'footer.php';?>

</body>
</html>
```

Résultat :

Welcome to my home page!

Some text.

Some more text.

Copyright © 2010-2022. isims.rnu.tn

Inclusion de fichiers externes
Activité

- Supposons que nous ayons un fichier de menu standard appelé "menu.html« , qui ressemble à ceci:

```
<a href="/default.php">Home</a> -
<a href="/html/default.php">HTML Tutorial</a> -
<a href="/css/default.php">CSS Tutorial</a> -
<a href="/js/default.php">JavaScript Tutorial</a> -
<a href="default.php">PHP Tutorial</a>
```

- Inclure le menu au début de la page « home.php »

Les super-globales
$_REQUEST, $_POST et $_GET

- $_REQUEST, $_POST et $_GET sont utilisés pour récupérer des données saisies après avoir soumettre un formulaire HTML.

```
<form method="post" action="<?=$_SERVER['PHP_SELF']?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>
```

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  // collect value of input field
  $name = $_REQUEST['fname'];
  if (empty($name)) {
    echo "Name is empty";
  } else {
    echo $name;
  }
}
?>
```

Exécution :

Les super-globales $_REQUEST, $_POST et $_GET

- **Attention !**
  - **$_REQUEST** peut récupérer les données saisies quelque soit la méthode d'envoi du formulaire (POST ou GET)
  - **$_POST** peut récupérer uniquement des données d'un formulaire soumis avec un méthode POST
  - **$_GET** peut récupérer uniquement des données d'un formulaire soumis avec un méthode GET

Les super-globales $_REQUEST, $_POST et $_GET

- **Activité:** soit le formulaire suivant

```
<form action="welcome_get.php" method="get">
Name: <input type="text" name="name"><br><br>
E-mail: <input type="text" name="email"><br><br>
<input type="submit">
</form>
```

  - Créez la page welcome_get.php, permettant de récupérer les données saisies dans le formulaire et de les afficher.
  - **Exemple d'exécution:**

Exercice

1. Créez un formulaire permettant à un membre d'une de ses associations de s'inscrire au repas, de donner ses disponibilités dans un ensemble de dates proposées et de spécifier quel type de plât il préparera.
   - La méthode d'envoi est POST
   - L'action est « inscription.php »

Exercice (suite)

2. Créez le script PHP « inscription.php » permettant d'afficher toutes les informations écrites dans le formulaire et de signaler une erreur si une information est manquante.