**SupGalilée info-1**

2023-2024

# Programmation Web
# Partie 3 : initiation à PHP

Étienne André

Université Sorbonne Paris Nord
Etienne.Andre@univ-paris13.fr



Version diapositives à trous : 29 avril 2024

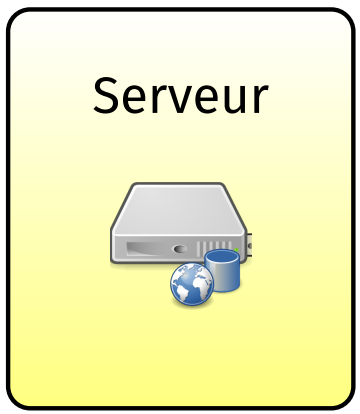
# Outline

# Motivation : Web dynamique

- Web statique : exclusivement HTML (et CSS)
  - Pages HTML stockées telles quelles sur le serveur
  - Rare ! (à part de tous petits sites Web)
- Problème des sites statiques :
  - Pas de base de données
  - Pas de personnalisation (requête, profil personnalisé, date…)
- Site Web dynamique : génération de pages HTML par un programme (ou script) côté serveur (*backend*)
  - Utilisation possible d'une base de données
  - Quelques langages pour le Web dynamique côté serveur : PHP, Ruby, ASP.NET, Node.JS…

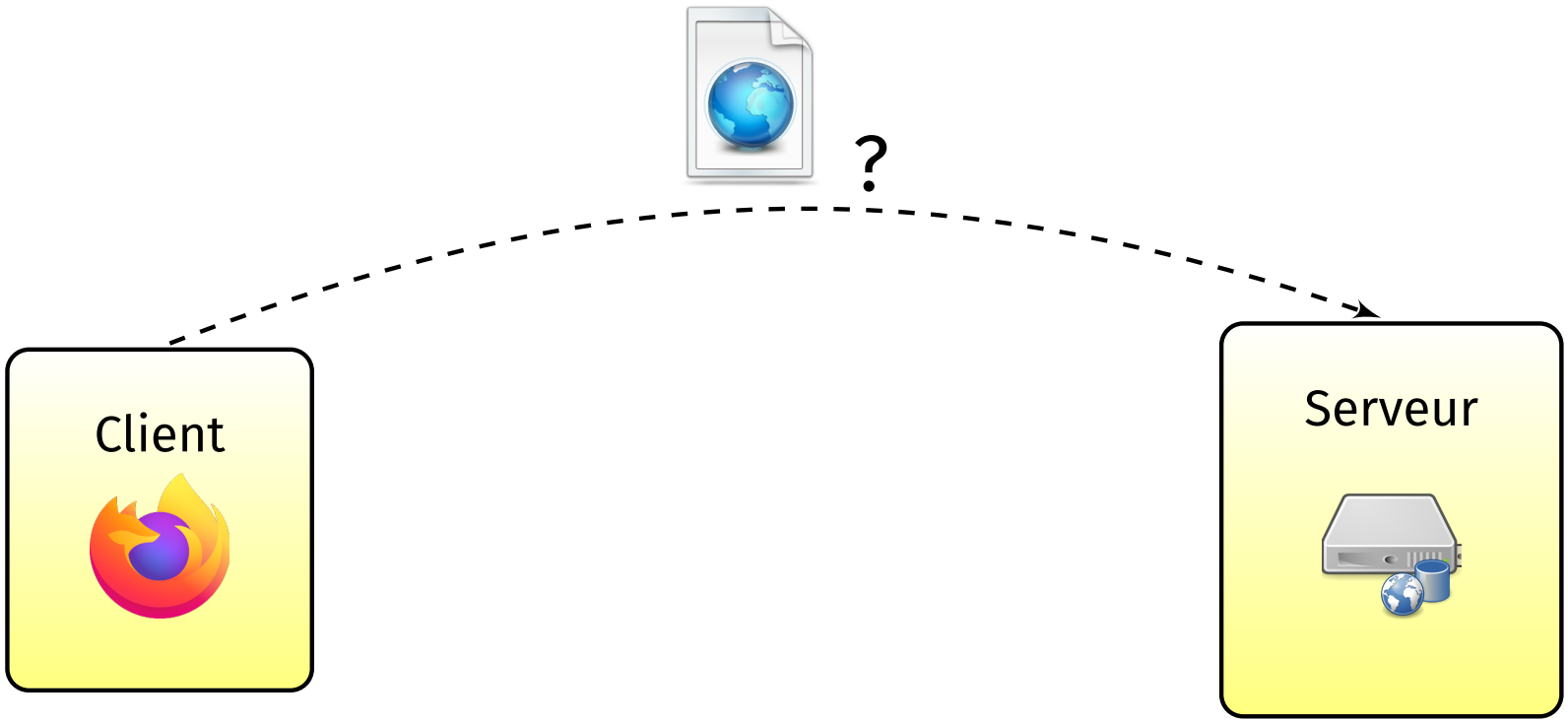
# Requête client-serveur (rappel du cours « client-serveur »)

Client :

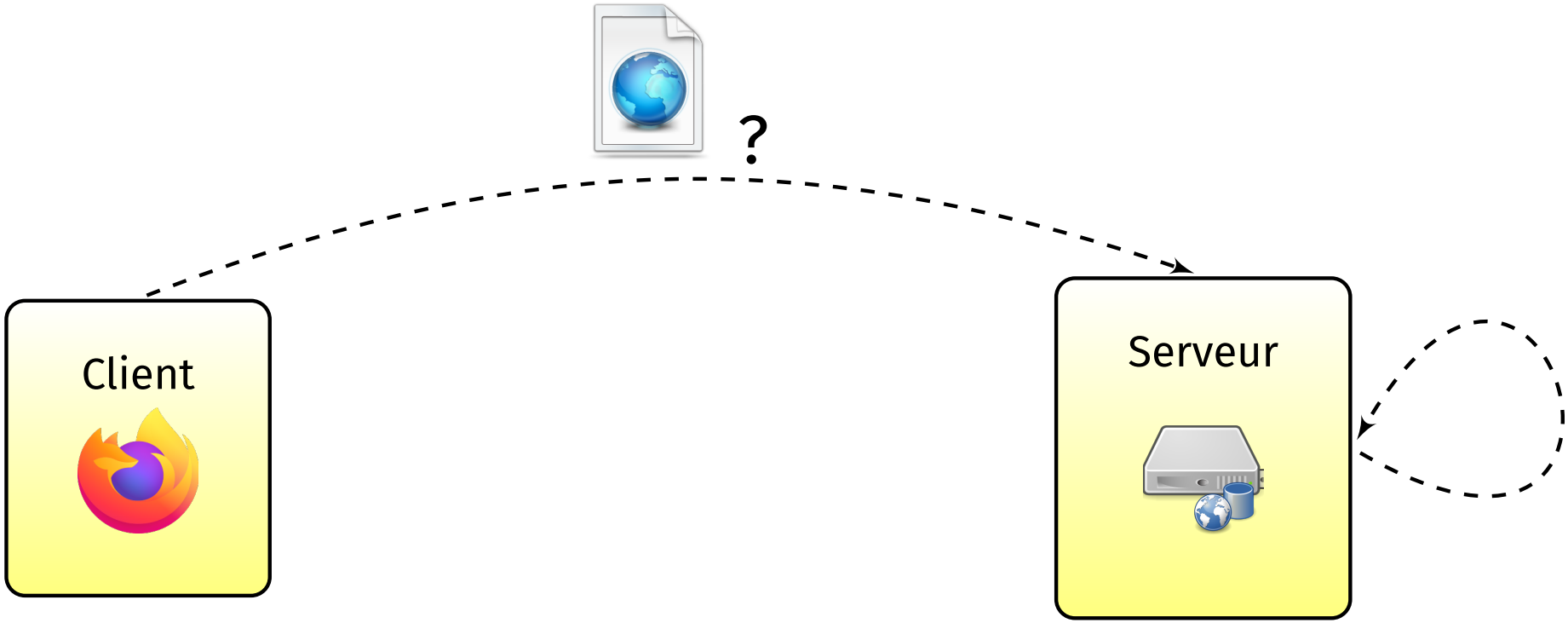
# Requête client-serveur (rappel du cours « client-serveur »)

Client : « Bonjour, je voudrais `https://www.univ-paris13.fr` »



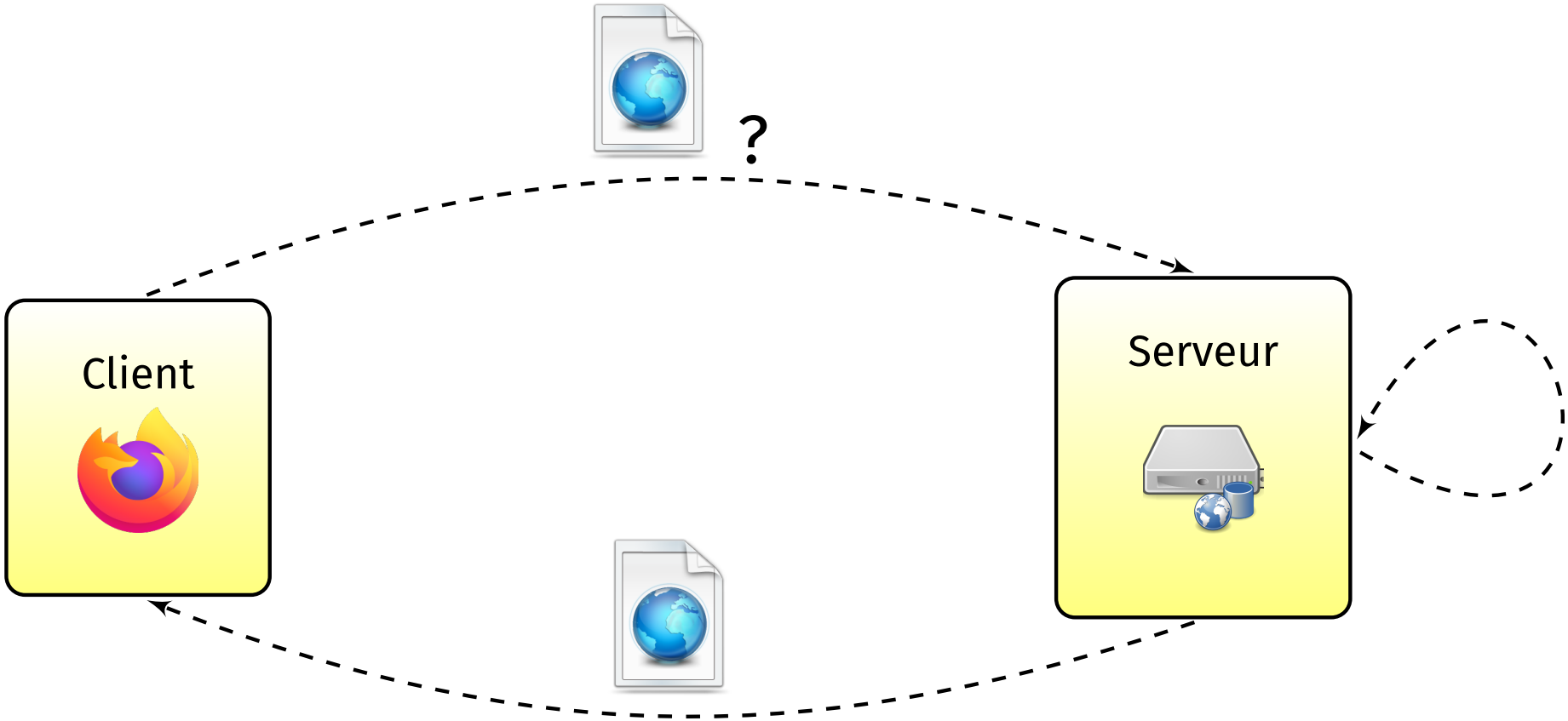
# Requête client-serveur (rappel du cours « client-serveur »)

Client : « Bonjour, je voudrais `https://www.univ-paris13.fr` »
(préparation de la page Web à l'aide d'un langage tel que PHP)



# Requête client-serveur (rappel du cours « client-serveur »)

Client : « Bonjour, je voudrais `https://www.univ-paris13.fr` »
(préparation de la page Web à l'aide d'un langage tel que PHP)
Serveur : « Bonjour, voici la page Web »



# Introduction

- PHP = PHP : Hypertext Preprocessor


☺ Acronyme récursif ! (ou sigle autoréférentiel ou sigle récursif)
  - Autres exemples : Bing, GNU, Visa...

# Historique

: Première version par Rasmus Lerdorf (à l'origine : une bibliothèque logicielle en C)

: réimplémentation complète par deux étudiants (Andi Gutmans et Zeev Suraski)

- 2015 : sortie de la version 7
- 2020 : sortie de la version 8
- 2022 : utilisé sur 78 % des serveurs Web

Source : `https://w3techs.com/`

# Historique

: Première version par Rasmus Lerdorf (à l'origine : une bibliothèque logicielle en C)

: réimplémentation complète par deux étudiants (Andi Gutmans et Zeev Suraski)

- 2015 : sortie de la version 7
- 2020 : sortie de la version 8
- 2022 : utilisé sur 78 % des serveurs Web

Source : `https://w3techs.com/`

# Principales caractéristiques (1/2)

- Langage impératif
- Langage orienté objet (présence de classes, interfaces et objets)
- Langage faiblement typé
- Langage sous licence libre

# Principales caractéristiques (2/2)

- Fichier avec extension `.php`
  - Peut être entièrement écrit en PHP, ou en HTML avec code PHP imbriqué
- Exécution entièrement côté serveur : pas accessible depuis le client (qui ne reçoit que la page HTML résultat)
- Nécessité d'une application côté serveur (exemple : serveur Apache ou nginx)


# PHP et le Web

- PHP est surtout utilisé dans un environnement de développement Web
  - Tourne sur un

# PHP et le Web

- PHP est surtout utilisé dans un environnement de développement Web
  - Tourne sur un

- PHP peut être notamment utilisé pour générer de façon dynamique
  - des pages HTML
  - des feuilles de style CSS
  - des scripts JavaScript

# Outline

# Outline

# Bref aperçu

Syntaxe relativement inspirée du langage C

```php
<?php
  $films = array('Taxi Driver', 'Spring Breakers',
    'Parasite', '12 Monkeys');
  echo "\n" . '<ul>';
  foreach($films as $film) {
    echo "\n\t" . '<li>' . $film . '</li>';
  }
  echo "\n" . '</ul>';
?>
```

... mais aussi beaucoup plus puissante et haut niveau que le C!

# Balises

Code à insérer entre <?php et ?>

```php
<?php
  $langage = 'PHP';
  echo '<p>Le ' . $langage . ", c'est bien !</p>";
  // Affiche "Le PHP, c'est bien !"
?>
```

# Interactions entre PHP et HTML

☹ Option 1 (non recommandée et dépréciée) : insertion dans le code HTML

```
<p>Nous sommes le <script language="php">echo date('d-m-y
    h:i:s');</script></p>
```

☹ Option 2 (pas très recommandée) : insertion dans le code HTML

```
<p>Nous sommes le <?php echo date('d-m-y h:i:s');?></p>
```

☺ Option 3 (meilleure) : génération de l'intégralité du HTML

```
<?php
  $aujourdhui = date('d-m-Y h:i:s');
  echo "<p>Nous sommes le" . $aujourdhui . '</p>';
?>
```

# Commentaires

Trois syntaxes pour les commentaires :

- Syntaxe dite « à la C » : bloc entre /* et */
- Fin d'une ligne après //
- Fin d'une ligne après # (syntaxe dite à la Bash ; moins utilisée ?)

```php
<?php
  $langage = 'PHP'; // Définition d'une variable
  /* $langage = 'OCaml';
  echo 'Je déteste le !' . $langage . '!';
  */
  echo 'Le ' . $langage . ", c'est bien !";
  # Affiche "Le PHP, c'est bien !"
?>
```

☺ PHP est exécuté côté serveur : les commentaires sont invisibles depuis le client

# Activation et désactivation des erreurs

En mode développement, il est recommandé d'activer les erreurs

- Affichage de messages d'erreur dans la page Web

```php
<?php
  // Activation de toutes les erreurs
  error_reporting(E_ALL);
?>
```

- (ces réglages peuvent être malgré tout désactivés par la configuration du serveur)

En mode production, il est fortement recommandé de désactiver les erreurs

- Raison : sécurité

```php
<?php
  // Désactivation de toutes les erreurs
  error_reporting(0);
?>
```

# Outline

# Les variables

- Syntaxe des noms de variables : $\$(\textit{lettre} \mid \_)(\textit{lettre} \mid \textit{chiffre} \mid \_)^*$

```php
// Exemples de déclarations de 3 variables différentes
$film  = 2046;
$_film = 'The Assassin';
$Film  = "Taxi Driver";
```

- Sensible à la casse (`$film` $\neq$ `$Film`)

- Typage dynamique

```php
// Déclarations successives
$film = 2046;
$film = 'The Assassin';
$film = false;
$film = array(2046, 'Spring Breakers', "Burning");
```

# Les nombres

- Les entiers

```php
// Nombres entiers
$a = 2046; $b = 300;
echo 2046 - 300;
echo intdiv(2046, 300); // affiche 6 (divison entière)
```

- Les flottants

```php
// Nombres flottants
$a = 2.046; $b = 2.04e6;
echo $a+$b; echo $a/$b;
```

- Nombreuses fonctions autour des nombres

⚠ Comme dans la plupart des langages, les nombres sont sujets au débordement et aux erreurs d'arrondi (`float`)

# Les chaînes de caractères : guillemets

Deux syntaxes principales :

- Guillemets simples : aucun nom de variable interprété

```php
echo 'J\'aime "Pour quelques $dollars de plus"\n';
// affiche: J'aime "Pour quelques $dollars de plus"\n
```

- Guillemets doubles : variables et retours à la ligne interprétés

```php
$film = 2046;
echo "J'aime le film \"$film\"\n";
// affiche: 'J'aime le film "2046"' suivi d'un retour à
    la ligne

echo "\t J'aime le film \"${film}\"";
//  affiche: '  J'aime le film "2046"'
```

En cas de doute : guillemets simples

# Les chaînes de caractères : affichage

Concaténation : opérateur « . »

Affichage d'une chaîne : `echo`

- On affiche généralement du code HTML !

```php
<?php
  $aujourdhui = date('d-m-Y h:i:s');
  echo "\n" . '<p>Nous sommes le' . $aujourdhui . '</p>';
  // affiche '<p>Nous sommes le ... </p>' (avec la date du
    jour)
?>
```

# Les variables externes (super-globales)

| Super-globale | Description | Exemple |
|---|---|---|
| `$GLOBALS` | Toutes les variables globales | `$GLOBALS['ma_variable']` |
| `$_POST` | Les variables envoyées par un formulaire (méthode `POST`) | `$_POST['mot_de_passe']` |
| `$_GET` | Les variables passées en paramètres à l'URL, ou par un formulaire (méthode `GET`) | `$_GET['page_demandee']` |
| `$_FILES` | Tableau associatif des valeurs associées aux documents téléchargés par le script courant | `$_FILES['nom_fichier']['size']` |
| `$_SESSION` | Tableau associatif des valeurs associées à la session courante | `$_SESSION['login']` |
| `$_COOKIE` | Tableau associatif des valeurs associées à un cookie | `$_COOKIE['internaute']` |
| `$_SERVER` | Informations liées au serveur utilisé | `$_SERVER['SERVER_PORT']` |

# Outline

# Structures de contrôle : if

Conditionnelle :

```php
$film = 'Spring Breakers'; // Définition d'une variable
if($film == 'The Assassin'){
  echo 'Meilleur film de la décennie 2010s';
}else{
  echo 'Pas le meilleur film';
}
// Affiche 'Pas le meilleur film'
```

# Structures de contrôle : boucle `while`

Boucle :

```php
$film = 0;
while($film < 2046){
  $film ++;
}
echo $film; // affiche 2046
```

# Structures de contrôle : boucle `do while`

Boucle :

```php
$film = 2046;
do{
  echo $film;
}while($film <> 2046);
// affiche 2046
```

Note : le corps est toujours exécuté au moins une fois, même si la condition est initialement fausse !

# Structures de contrôle : boucle `for`

Itération avec condition :

```php
$films = array('Taxi Driver', 'Spring Breakers',
    'Parasite', '12 Monkeys');
for($i = 0; $i < count($films); $i++) {
    echo $films[$i] . "\n";
}
```

# Structures de contrôle : boucle `foreach`

Itération sur un ensemble :

```php
$films = array('Taxi Driver', 'Spring Breakers',
    'Parasite', '12 Monkeys');
foreach($films as $film) {
    echo $film . "\n";
}
```

# Structures de contrôle : case

Traitement par cas :

```php
switch ($film) {
  case 'Spring Breakers':
  case '12 Monkeys':
    echo 'Film américain';
    break;
  case 'Tigre et Dragon':
    echo 'Film taïwanais';
    break;
  default:
    echo 'Film inconnu';
}
```

⚠ Ne pas oublier le `break`; sinon les instructions suivantes sont exécutées

# Outline

# Les tableaux en PHP

- À mi-chemin entre deux concepts de programmation :
  - un tableau (valeurs avec indices croissants), et
  - une table associative (ensemble de couples (clé, valeur))

- Caractéristiques
  - Pas de taille prédéfinie
  - Pas de type : peuvent contenir (presque) n'importe quoi

- Syntaxe
  - Syntaxe de déclaration :
    - `array()`, ou
    - `[]`
  - Association d'une valeur à une clé :
    - `clé => valeur`, ou
    - `$montableau[cle] = valeur`

# Les tableaux en PHP : exemple

```php
// Ensemble de valeurs
$realisatrices = array('Ducournau', 'Despentes', 'Sciamma');
echo $realisatrices[1]; // Affiche 'Despentes'

// Clés => valeurs
$films = array(
    'Petite Maman' => array(
        'real' => 'Sciamma',
        'sortie' => 2021,
        'interpretation' => array('Sanz', 'Meurisse')
    ),
    'Jackie Brown' => array(
        'real' => 'Tarantino',
        'sortie' => 1997,
        'interpretation' => array('Grier', 'Jackson', 'De Niro')
    ),
    'Lalaland'     => 'aucune info'
);
echo $films['Jackie Brown']['interpretation'][2]; // Affiche 'De Niro'
```

# Outline

# Les fonctions

- Fonctions définies par le langage (*built-in*) : plus de 1000 !
- Fonctions définies dans le programme : définies par le mot-clé `function`

```php
function affiche_film($titre, $annee) {
  echo "\n<p>$titre a été réalisé en $annee.</p>";
}

affiche_film('Taxi Driver',     1976);
affiche_film('Tigre et Dragon', 2000);
affiche_film('Dunkerque',       2017);
// <p>Taxi Driver a été réalisé en 1976.</p>
// <p>Tigre et Dragon a été réalisé en 2000.</p>
// <p>Dunkerque a été réalisé en 2017.</p>
```

# Mode strict

Avec PHP 7, un peu plus de typage a été introduit avec le mode `strict`

- Déclaration

```php
<?php
  // Première instruction du fichier
  declare(strict_types=1);
?>
```

- Conséquence : typage optionnel des fonctions (mais doit alors être valide !)
  - Sinon : exception (`Fatal error: Uncaught TypeError`)

```php
function somme(int $a, int $b) : int{
  return $a + $b;
}
function affiche_film(string $titre, int $annee) : void{
  echo "\n<p>$titre a été réalisé en $annee.</p>";
}
```

# Valeurs des arguments par défaut

Si un argument n'est pas spécifié, la valeur par défaut est utilisée

Exemple :

```php
function affiche_film(string $titre, int $annee = 2024) {
  echo "\n<p>$titre a été réalisé en $annee.</p>";
}
affiche_film('Taxi Driver',     1976);
affiche_film('Tigre et Dragon', 2000);
affiche_film('Dunkerque');
// <p>Taxi Driver a été réalisé en 1976.</p>
// <p>Tigre et Dragon a été réalisé en 1976.</p>
// <p>Dunkerque a été réalisé en 2024.</p>
```

# Passage par référence

Il est possible de passer un argument par référence et non par valeur : sa valeur est donc modifiée, y compris à l'extérieur de la fonction

```php
// Passage par valeur
function increment(int $n){
  $n++;
}
// Passage par référence (syntaxe &)
function increment2(int &$n){
  $n++;
}
$a = 12; $b = 2045;
increment($a);
echo $a; // affiche 12

increment2($b);
echo $b; // affiche 2046
```

# Tableaux en arguments

⚠ Attention : les tableaux sont aussi passés par valeur (et non par référence)

- Si le contenu d'un tableau est modifié dans une fonction où il est passé en argument, ça ne s'applique pas à l'extérieur

```php
$montableau = [0, 1, 2, 3];

function modiftableau($tableau){
  $tableau[1] = 2046;
  echo $tableau[1]; // affiche 2046
}

echo $montableau[1]; // affiche 1
modiftableau($montableau);
echo $montableau[1]; // affiche 1
```

Différent d'autres langages !

# Portée des variables dans les fonctions

- Variable locale : variable définie dans une fonction
- Si une variable locale possède le même nom qu'une variable globale, la variable locale est prioritaire
  - La variable globale peut être accédée par le tableau de variables super-globales prédéfini `$GLOBAL['nom_variable]'` ; ou en déclarant la variable avec le mot-clé `global`

```
1  $a = 300;
2  $b = 2010;
3  $c = 2045;
4  function test_var_globales(){
5    global $b;
6    echo $a; // n'affiche rien !
7    $b += 2;
8    $GLOBALS['c'] ++;
9  }
10 test_var_globales();
11 echo $b; // affiche 2012
12 echo $c; // affiche 2046
```

# Outline

# PHP et la programmation orientée objet

Quelques fonctionnalités orientées objet en PHP :

- Classes
- Interfaces
- Classes anonymes
- Objets
- Héritage (simple)
- Méthodes statiques
- Visibilité différenciée (`public`, `protected`, `private`)

# Mini-initiation à l'objet en PHP via un exemple

```php
class Film{
    protected $titre = ''; // accessible dans les sous-classes
    function __construct($titre) {
        $this->titre = $titre;
    }
    public function affiche(){
        echo 'Le film ' . $this->titre . ' est bien';
    }
}

class Chefdoeuvre extends Film{ // heritage
    public function affiche(){ // redéfinition ('overriding')
        echo 'Le film ' . $this->titre . ' est génial';
    }
}
$film1 = new Film('Kill Bill');
$film2 = new Chefdoeuvre('Jackie Brown');
$film1->affiche(); // 'Le film Kill Bill est bien'
$film2->affiche(); // 'Le film Jackie Brown est génial'
```

# Outline

# Génération de code HTML

PHP est généralement utilisé pour générer tout ou partie du code HTML

- et/ou CSS, et/ou JavaScript

Exemple de code PHP :

```php
$films = array('Taxi Driver', 'Spring Breakers',
    'Parasite', '12 Monkeys');
echo "\n" . '<ul>';
foreach($films as $film) {
    echo "\n\t" . '<li>' . $film . '</li>';
    echo "\n\t<li>$film</li>"; // variante avec guillemets
    doubles
}
echo "\n" . '</ul>';
```

Génère le code HTML suivant :

```html
<ul>
  <li>Taxi Driver</li>
  <li>Spring Breakers</li>
  <li>Parasite</li>
  <li>12 Monkeys</li>
</ul>
```

# Génération de code HTML : attention aux espaces

Ne pas oublier qu'il y a trois niveaux :

- le code PHP
- le code HTML généré par PHP
- et le rendu de la page Web spécifié par le code HTML

Code PHP :

```php
echo "\n\t" . '<p>Un';
echo ' exemple' . "\n" . ' d\'espacement</p><p>bizarre' .
    "\n" . '</p>';
```

Code HTML généré

```
<p>Un exemple
d'espacement</p><p>bizarre
</p>
```

Rendu sur le navigateur :

Un exemple d'espacement
bizarre

# Outline

# La méthode de requête HTTP « GET »

Permet notamment de :

1. Récupérer les valeurs passées en paramètre à l'URL
   - Exemple : `https://www.startpage.com/sp/search?query=get+php&cat=web&pl=opensearch&language=francais`
   - Massivement utilisé aujourd'hui : regardez les URL des sites que vous visitez!

2. Récupérer les informations transmises par un formulaire via la méthode `get`
   - `method="get"`

# Get et les paramètres de l'URL

- Paramètres de l'URL : ensemble de couples (clé, valeur)
- Syntaxe : `https://www.monsite.fr?cle1=valeur1&cle2=valeur2&cle3=valeur3&...`
- Exemple :

  `https://www.startpage.com/sp/search?query=get+php&cat=web&pl=opensearch&language=francais`

  - (`query`, `get+php`)
  - (`cat`, `web`)
  - (`pl`, `opensearch`)
  - (`language`, `francais`)
- Attention aux caractères spéciaux
  - Certains caractères (« / », « & », espaces...) sont proscrits dans les clés et les valeurs
  - Utiliser des fonctions de conversion automatique (`urlencode`)

# Get et PHP

Récupération des arguments : variable PHP prédéfinie `$_GET`

Exemple :

- URL : `https://www.monsite.fr/index.php?film=Titane&annee=2021`
- Code PHP et affichage :

```php
echo '<p>Le film ' . $_GET['film'] . ' a été réalisé en ' .
    $_GET['annee'] . ' !</p>';
// Si l'on a cliqué sur l'URL ci-dessus, va afficher 'Le
    film Titane a été réalisé en 2021 !'
```

# Get et PHP : caractères spéciaux

- Pour encoder une URL (en échappant les espaces, notamment), utiliser `urlencode`
- Les valeurs récupérées dans `$_GET` sont déjà décodées (pas besoin de convertir les caractères spéciaux !)

# Get et PHP : caractères spéciaux

- Pour encoder une URL (en échappant les espaces, notamment), utiliser `urlencode`
- Les valeurs récupérées dans `$_GET` sont déjà décodées (pas besoin de convertir les caractères spéciaux !)

Exemple d'encodage d'URL :

```php
echo '<a href="' . $_SERVER['PHP_SELF'] . '?cours=' .
    urlencode('Prog Web') . '&prof=' . urlencode('Étienne
    André') . '">Cliquer ici</a>';
```

Va afficher un lien vers

`https://www.monsite.fr/index.php?cours=Prog+Web&prof=%C3%89tienne+Andr%C3%A9`

Exemple de récupération des paramètres de l'URL :

```php
echo '<p>Bonjour ' . $_GET['prof'] . ', prof de ' .
    $_GET['cours'] . ' !</p>';
// Si l'on a cliqué sur l'URL ci-dessus, va afficher
    'Bonjour Étienne André, prof de Prog Web !'
```

# Formulaires (méthode get) et PHP : petit exemple (1/2)

## Formulaire HTML

```
<form action="<?php echo $_SERVER['PHP_SELF'];?>"
    method="get">
     <p><label for="idfilm">Film</label>
       <input type="text" name="film" id="idfilm"
    placeholder="Votre meilleur film"></p>
</form>
```

## Récupération et traitement des valeurs (PHP)

```
if ($_SERVER["REQUEST_METHOD"] == "GET") {
  // Récupération des données
  $film = $_GET['film']; // `film` = `name` du form
  if (empty($film)) {
    echo "<strong>Erreur</strong> : pas de film";
  } else {
    echo $film;
  }
}
```

... ces deux codes peuvent être sur la même page (même URL)!

# Formulaires (méthode `get`) et PHP : petit exemple (2/2)

La conversion est automatique à l'encodage et au décodage

Exemple :

- Si l'on entre dans le champ `nom` :
  « `Le /département/ R&T ?!` »
- Alors l'URL générée est :

  `https://www.monsite.fr/index.php?nom=Le+%2Fd%C3%A9partement%2F+R%26T+%3F%21`
- Et, après récupération via PHP, le texte affiché en HTML est :
  « `Le /département/ R&T ?!` »

# Formulaires et PHP : discussion

La même page peut être utilisée pour afficher le formulaire et le traitement

- Possibilité de ne pas réafficher le formulaire une fois que les données ont été envoyées et traitées
- En cas d'erreur et de rechargement de la page avec le même formulaire, les informations entrées sont par défaut perdues
  - possibilité de réafficher les informations envoyées (pour pré-remplir le formulaire) grâce aux sessions (voir plus loin)

Complémentaire à JavaScript

- JavaScript peut effectuer des vérifications avant l'envoi du formulaire
- Attention néanmoins : JavaScript étant exécuté côté client, un client néfaste peut décider de désactiver voire modifier les scripts JavaScript $\Rightarrow$ ne pas baser toutes les vérifications sur JavaScript !

# Outline

# La méthode de requête HTTP « POST »

- PHP peut récupérer et traiter (notamment) les valeurs associées à un formulaire ayant utilisé la méthode `post`
  - Via la variable PHP prédéfinie `$_POST`

# Formulaires (méthode post) et PHP : petit exemple

Exemple (HTML)

```
<form action="<?php echo $_SERVER['PHP_SELF'];?>"
    method="post">
     <p><label for="idfilm">Film</label>
       <input type="text" name="film" id="idfilm"
    placeholder="Votre film préféré"></p>
</form>
```

Récupération et traitement des valeurs (PHP)

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  // Récupération des données
  $film = $_POST['film']; // `film` = `name` du form
  if (empty($film)) {
    echo "<strong>Erreur</strong> : pas de film";
  } else {
    echo $film;
  }
}
```

... ces deux codes peuvent être sur la même page (même URL)!

# Outline

# Cookies

Cookie : format texte constitué de couples (clé, valeur) échangé avec le serveur, et stocké côté client

- peut stocker des informations telles que des préférences de l'internaute, ou son identifiant (permettant le suivi par le site Web)
- peut avoir une durée de vie de plusieurs années !
- peut être supprimé aisément (via le navigateur)
- utilisé (surtout) pour pister les internautes sur un même site, voire sur plusieurs sites différents

Des bons cookies (c'est pas comme d'autres)

# Les cookies en PHP

Création d'un cookie :

```php
<?php
  if(!isset($_COOKIE['id_internaute'])){
      setcookie('id_internaute', '12345');
      // 12345 est généralement un identifiant unique
  }
?>
```

Accès à un cookie :

```php
<?php
  if(isset($_COOKIE['id_internaute'])){
    echo 'Identifiant de session : ' .
    $_COOKIE['id_internaute'];
  }
?>
```

permet de retrouver l'internaute même des années après! (depuis le même navigateur)

# Durée de vie d'un cookie

Syntaxe : `setcookie(cle, valeur, date_expiration)`

```php
setcookie('id', '12345', time()+365*24*3600);  // un an
setcookie('pref', 'site_mobile', time()+3600); // une heure
```

Supprimer un cookie : en indiquant une date passée

```php
setcookie('id', '', time()-3600); // date passée !
```

# Que stocker dans un cookie ?

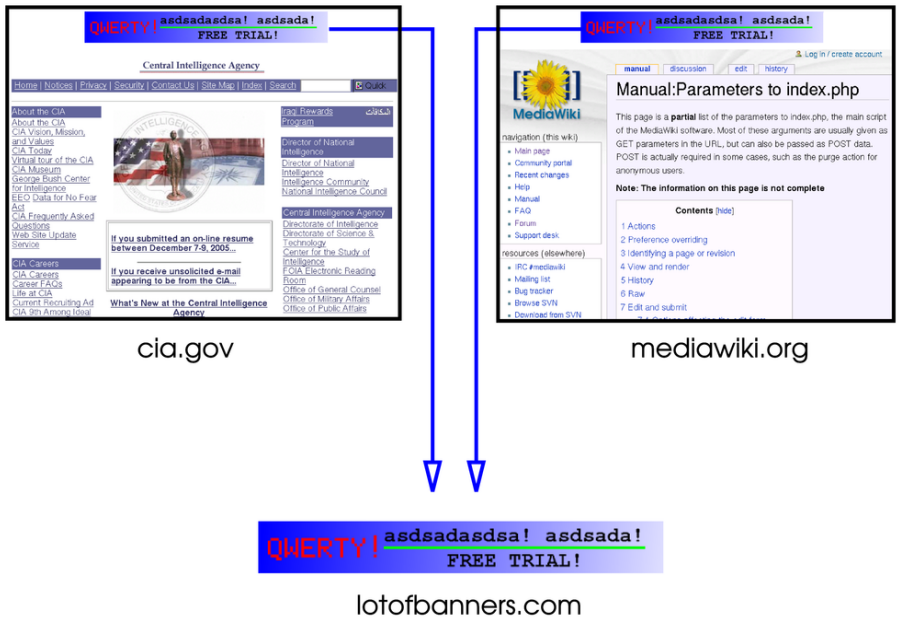
- identifiant de l'internaute (permet de rester connecté·e)
- préférence de navigation
- contenu du panier sur un site de commerce (de plus en plus remplacé par une base de données à partir de l'identifiant de l'internaute)
  - en réalité : origine des cookies !
- etc.

# Danger des cookies

- Identification de visites répétées à un site même sans être identifié·e
  - le site garde en mémoire toutes les pages visitées par ce même identifiant « anonyme »
- Croisement avec votre véritable identité si l'on s'identifie une seule fois au site
  - le site ajoute à votre historique de navigation (stocké dans sa base de données) l'intégralité des visites qu'il a tracées via votre identifiant anonyme : fusion de l'historique de votre identifiant anonyme avec l'historique de votre identifiant de connexion
- Menace sur la vie privée… et bien plus encore en cas de cookies tierce partie

# Les cookies tierce partie

Cookie tierce partie : déposé ou lu par un autre site que celui sur lequel l'Internaute est connecté·e




Exemples d'utilisations massives :

- Google Analytics
- Publicités Google (ou autres)
- Bouton « Like » : Facebook peut ainsi vous suivre sur (presque) tout Internet sans que vous ne vous rendiez jamais sur le site de Facebook!

# Les cookies tierce partie : résistances

- De nombreux navigateurs acceptent les cookies tierce partie par défaut, mais ils peuvent être désactivés via les préférences du navigateur
- Google a cessé d'utiliser les cookies tierce partie en 2022
  - … mais il existe de nombreuses autres façons d'identifier un·e Internaute de façon (quasi) unique
- Facebook n'affiche plus les boutons « J'aime » pour les internautes d'Europe, sauf si elles/ils sont connecté·e·s à leur compte et ont donné leur consentement

`https://developers.facebook.com/docs/plugins/like-button/?locale=fr_FR`

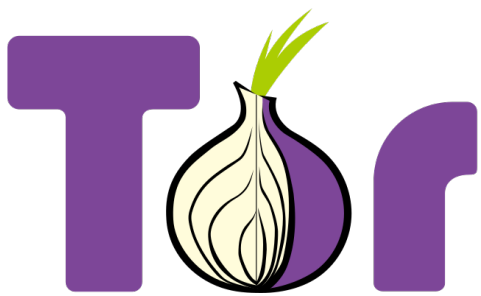
# Danger des cookies et navigation privée

La navigation privée permet (notamment) de supprimer les cookies à la fin d'une navigation

- Le navigateur supprime tout (cookies, historique…) en fin de navigation
- N'empêche pas entièrement le traçage par les sites Web pour autant!

## ⚠ Limites de la navigation privée

La navigation privée n'empêche pas du tout votre fournisseur d'accès (et donc toute entité étatique) de connaître les sites que vous visitez!
Seule l'utilisation d'un réseau tel que TOR permet (a priori) une navigation véritablement privée.



# Outline

# Sessions

Session en PHP :

- période de temps où un serveur est en communication authentifiée avec un·e internaute
- une session ne concerne qu'un·e seul·e internaute à la fois
- durée variable, typiquement une heure
- contient habituellement des informations liées au compte d'un·e internaute
- permet de faire persister des informations entre plusieurs pages (du même serveur)

## Pas d'infos confidentielles

Les informations confidentielles (par exemple authentification par mot de passe, numéro de carte bancaire…) ne doivent pas être conservées au sein d'une session

# Utilité des sessions

- Mémoriser temporairement des informations de l'internaute
- Pré-remplir un formulaire avec les informations précédemment envoyées (par exemple : en cas d'erreur de saisie et réaffichage du même formulaire)
- Utilisation en conjonction avec les cookies

# Sessions : exemple simple

```php
<?php
// Nécessaire pour utiliser les sessions
session_start();

// Création d'un compteur de visites (de la MÊME personne)
if (!isset($_SESSION['nb_visites'])) {
   $_SESSION['nb_visites'] = 0;
} else {
   $_SESSION['nb_visites']++;
}
echo $_SESSION['nb_visites'];
?>
```

Comportement :

- le compteur va s'incrémenter à chaque rechargement de la page
- une autre personne (ou soi-même sur un différent navigateur, ou le même navigateur en navigation privée) aura sa propre valeur du compteur !

# Sessions : quelques fonctions utiles

- Activation des sessions (avant toute génération de code HTML)

```php
session_start();
```

- Suppression de toutes les variables de session

```php
session_unset();
```

- Fermeture de la session en cours

```php
session_destroy();
```

# Destruction de session après une durée donnée

```php
session_start();
if (isset($_SESSION['debut']) && (time() -
    $_SESSION['debut'] > 1800)) {
    session_unset();
    session_destroy();
    echo "session terminée";
}
$_SESSION['debut'] = time();
```

Comportement :

- la session est supprimée après 30 minutes sans rechargement
- ... mais elle est sinon automatiquement prolongée de 30 minutes à chaque rechargement

# Destruction de session après une durée donnée

```php
session_start();
if (isset($_SESSION['debut']) && (time() -
    $_SESSION['debut'] > 1800)) {
    session_unset();
    session_destroy();
    echo "session terminée";
}
$_SESSION['debut'] = time();
```

Comportement :

- la session est supprimée après 30 minutes sans rechargement
- ... mais elle est sinon automatiquement prolongée de 30 minutes à chaque rechargement

Syntaxe alternative :

```php
session_regenerate_id(true);
```

# Outline

# Motivation

- PHP permet de générer des pages Web dynamiques… mais ne permet pas (particulièrement) de stocker de l'information

- Solution : base de données
  - Base de recherche des moteurs de recherche
  - Posts et commentaires sur les réseaux sociaux
  - Stockage d'identifiants personnels
  - Stockage de l'historique de navigation d'un·e internaute par une régie publicitaire ou un réseau social
  - Courriels
  - etc.

# MySQL

- Système de gestion de bases de données relationnelles
- Première version :

# MySQL

- Système de gestion de bases de données relationnelles
- Première version :
- Distribué sous licence libre (et propriétaire)
- Très efficace en lecture (un peu moins en écriture)
- Concurrents de MySQL : Microsoft SQL server, Oracle, PostgreSQL

# MySQL et PHP

- Couple PHP-MySQL très fréquent (mais pas exclusif !)
- Utilisé notamment au sein de l'ensemble LAMP
  - Linux, Apache, MySQL, PHP (ou Perl/Python)
  - Variantes / concurrents : LAMA, MAMP, WAMP…
- PHP offre un ensemble de fonctions en lien avec MySQL

# Outline

# Connexion à une base de données MySQL

## Syntaxe orientée objet (plus récente)

```php
$mysqli = new mysqli('serveur', 'user', 'mdp', 'base');
// NOTE: peut aussi prendre le port et le socket en arguments optionnels
if ($mysqli -> connect_error) {
    /* Gestion de l'erreur */
}
```

## Syntaxe procédurale (la syntaxe historique)

```php
$connexion = mysqli_connect('serveur', 'user', 'mdp', 'base');
if (!$connexion) {
    /* Gestion de l'erreur */
}
```

# Gestion des erreurs de connexion

Ne pas afficher les erreurs par défaut mais les gérer depuis PHP :

```php
/* Désactiver les rapports d'erreur */
mysqli_report(MYSQLI_REPORT_OFF);

/* Le `@` permet de supprimer les warnings */
$mysqli = @new mysqli('serveur', 'user', 'mdp', 'base');

if ($mysqli->connect_error) {
    /* Gestion de l'erreur */
    echo ('Erreur de connexion : ' .
    $mysqli->connect_error);
    exit(1);
}
```

# Requête MySQL en PHP

Syntaxe orientée objet :

```php
/* Requête pour récupérer 5 noms de films (table `Films`) */
$resultat = $mysqli->query("SELECT nom FROM Films LIMIT 5");
printf("%d résultats retournés.\n", $resultat->num_rows);
```

Syntaxe procédurale

```php
$resultat = mysqli_query($connexion, "SELECT nom FROM Films
    LIMIT 5");
printf("%d résultats retournés.\n",
    mysqli_num_rows($resultat));
```

# Exemple complet PHP et MySQL (orienté objet)

```php
$mysqli = new mysqli('serveur', 'user', 'mdp', 'base');
if ($mysqli -> connect_error) {
    /* ... Gestion de l'erreur ... */
}

$resultat = $mysqli->query("SELECT nom, annee FROM Films
    LIMIT 5");
printf("%d résultats retournés.\n", $resultat->num_rows);

while ($ligne = $resultat->fetch_assoc()) {
  echo $ligne['nom'] . ' a été réalisé en ' .
    $ligne['annee'];
}

$resultat -> free_result();

$mysqli -> close();
```

# Exemple complet PHP et MySQL (procédural)

```php
$connexion = mysqli_connect('serveur', 'user', 'mdp',
    'base');
if (!$connexion) {
    /* ... Gestion de l'erreur ... */
}

$resultat = mysqli_query($connexion, "SELECT nom, annee
    FROM Films LIMIT 5");
printf("%d résultats retournés.\n",
    mysqli_num_rows($resultat));

while ($ligne = mysqli_fetch_array($result)) {
  echo $ligne['nom'] . ' a été réalisé en ' .
    $ligne['annee'];
}

mysqli_free_result($resultat);

mysqli_close($connexion);
```

# Codages de caractÃ¨res

Attention, il faut dÃ©finir :

- un codage en PHP
- un codage en MySQL
- un codage en HTML

Le mieux : tout en UTF-8 !

Fonctions utiles en PHP :

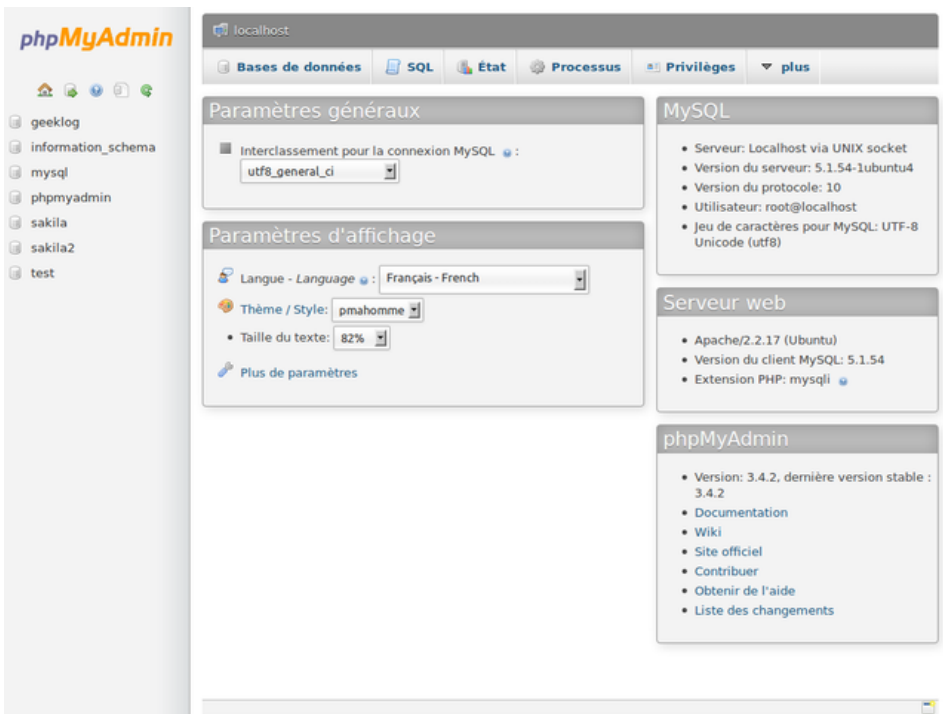
```php
/* Pour MySQL: définir la base en `utf8_unicode_ci` */

/* Puis préciser le codage lors de la connexion */
$connexion->set_charset("utf8");

/* Pour renvoyer un fichier (ex : HTML) au bon format */
header('Content-type: text/html; charset=utf-8');
```

# phpMyAdmin

- Application Web (en PHP) pour la gestion de bases MySQL
- Permet des requêtes (création de tables, export, accès…) de façon intuitive et graphique
- Logiciel distribué sous licence GNU GPL
- Souvent installé par défaut sur les serveurs et hébergeurs (par exemple dans le cadre d'une solution LAMP)



# Outline

# ⚠ Attention aux injections SQL !

## ⚠ Attaque par injection SQL

Toute interaction entre PHP et MySQL peut présenter un risque d'attaque par injection SQL.

Idée : l'attaquant·e va chercher à insérer du code dans un formulaire, qui sera ensuite utilisé pour effectuer des requêtes néfastes (obtention de droits admin, récupération d'informations confidentielles, destructions…).

# Injection SQL par commentaire

Code PHP :

```
1 $resultat = $mysqli->query("SELECT nom, annee FROM Films
    WHERE realisateur = '" . $_POST['realisateur'] . "'");
```

- ☺ Si l'attaquant·e entre « `Tarantino` » dans le formulaire :
  - La requête SQL est « `SELECT nom, annee FROM Films WHERE realisateur = 'Tarantino'` »

# Injection SQL par commentaire

Code PHP :

```php
$resultat = $mysqli->query("SELECT nom, annee FROM Films
    WHERE realisateur = '" . $_POST['realisateur'] . "'");
```

- ☺ Si l'attaquant·e entre « `Tarantino` » dans le formulaire :
  - La requête SQL est « `SELECT nom, annee FROM Films WHERE realisateur = 'Tarantino'` »
- ☹ Si l'attaquant·e entre « `Tarantino'; drop table Films; -- -` » dans le formulaire :
  - La requête SQL devient « `SELECT nom, annee FROM Films WHERE realisateur = 'Tarantino'; drop table Films; -- -'` »
    - Rappel : la syntaxe « `-- -` » en MySQL correspond aux commentaires : le reste est ignoré

# Injection SQL par disjonction (OR)

Code PHP :

```php
$resultat = $mysqli->query("SELECT id FROM users WHERE nom
    = '" . $_POST['nom'] . "' AND mdp = '" . $_POST['mdp']
    . "'");
```

- ☺ Si l'attaquant·e entre « anissa » et « MdP?1234@ » dans le formulaire :
  - La requête SQL est « `SELECT id FROM users WHERE nom = 'anissa' AND mdp = 'MdP?1234@'` »
  - Va fonctionner uniquement si « MdP?1234@ » est le bon mot de passe

# Injection SQL par disjonction (OR)

Code PHP :

```php
$resultat = $mysqli->query("SELECT id FROM users WHERE nom
    = '" . $_POST['nom'] . "' AND mdp = '" . $_POST['mdp']
    . "'");
```

- ☺ Si l'attaquant·e entre « anissa » et « MdP?1234@ » dans le formulaire :
  - La requête SQL est « `SELECT id FROM users WHERE nom = 'anissa' AND mdp = 'MdP?1234@'` »
  - Va fonctionner uniquement si « MdP?1234@ » est le bon mot de passe
- ☹ Si l'attaquant·e entre « admin » et « `hihi' OR 1; -- -` » dans le formulaire :
  - La requête SQL est « `SELECT id FROM users WHERE nom = 'admin' AND mdp = 'hihi' OR 1; -- -'` »
  - À cause du « `OR 1` », la 2e partie fonctionne toujours : l'attaquant·e obtient les droits de l'utilisatrice sans connaître son mot de passe !

# Éviter les injection SQL en PHP

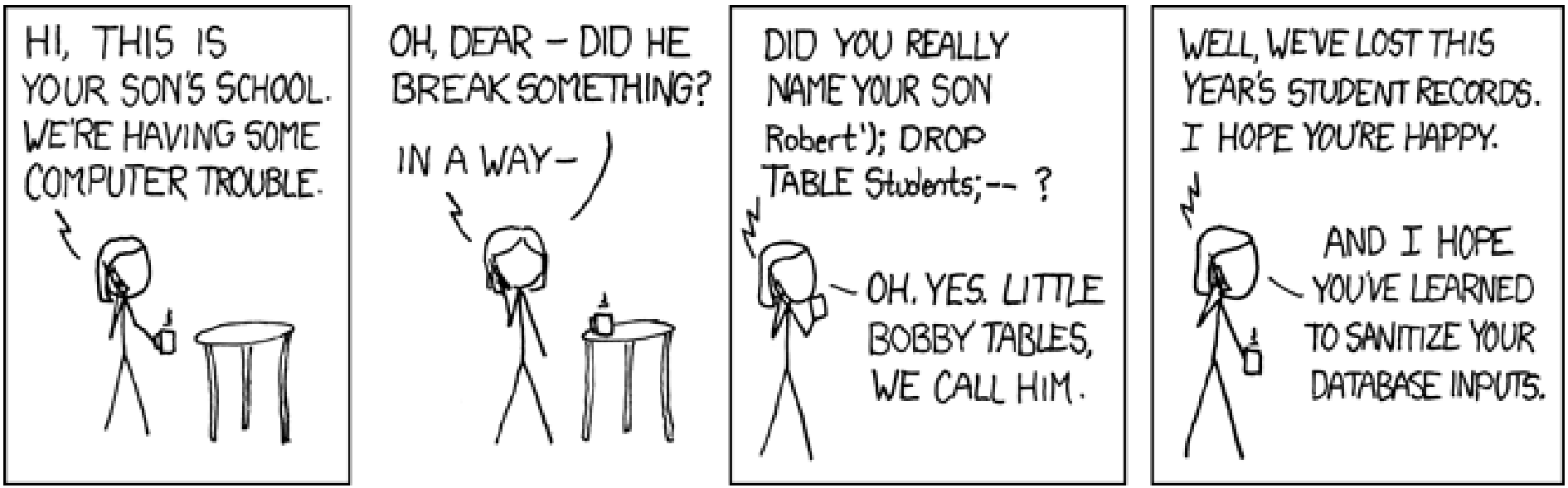
## Empêcher les injections SQL grâce aux fonctions PHP

(Début de) solution : utiliser la fonction `mysqli_real_escape_string` (ou `mysqli::real_escape_string` en syntaxe orientée objet)

```php
/* Version orientée objet */
$requete = sprintf("SELECT id FROM users WHERE nom = '%s'
    AND mdp = '%s'",
     $mysqli->real_escape_string($_POST['nom']),
     $mysqli->real_escape_string($_POST['mdp'])
);
$resultat = $mysqli->query($requete);

/* Version procédurale */
$requete = sprintf("SELECT id FROM users WHERE nom = '%s'
    AND mdp = '%s'",
     mysqli_real_escape_string($connexion, $_POST['nom']),
     mysqli_real_escape_string($connexion, $_POST['mdp'])
);
$resultat = mysqli_query($connexion, $requete);
```

# Les injections SQL vues par XKCD




Source : `https://xkcd.com/327/`

# Alternative aux fonctions PHP vues ici : PDO

PDO = *PHP Data Objects*

- extension de PHP
- interface d'accès aux bases de données depuis PHP
- trois classes : `PDO`, `PDOStatement`, `PDOException`
- gère la connexion à la base de données, l'envoi des requêtes, la déconnexion
- permet de changer plus facilement de système de gestion de bases de données

Voir `https://www.php.net/manual/fr/book.pdo.php`

# Sources et références

# Sources et références

## Références

- Robin Nixon : Développer un site web en PHP, MySQL JavaScript jQuery, CSS3 et HTML5 : Un guide étape par étape pour créer des sites Web dynamiques (O'Reilly)

## Sources

- Ce cours a bénéficié d'améliorations tirées des supports de cours de Thamer Mecharnia (Université Sorbonne Paris Nord)

# Crédits

# Source des images utilisées I

Titre : warning sign
Auteur : ?
Source : https://commons.wikimedia.org/wiki/File:Warning_sign.png
Licence : PUBLIC DOMAIN

Titre : Check Mark CSS Green
Auteur : Pigeon43
Source : https://commons.wikimedia.org/wiki/File:Check_Mark_CSS_Green.svg
Licence : CC BY

Titre : The logo of Firefox
Auteur : Mozilla Corporation
Source : https://commons.wikimedia.org/wiki/File:Firefox_logo,_2019.svg
Licence : moz://a

Titre : Server-web-database
Auteur : RRZEicons
Source : https://commons.wikimedia.org/wiki/File:Server-web-database.svg
Licence : CC BY SA

Titre : Vista Icons Toolbar
Auteur : VistaICO.com

# Source des images utilisées II

Source : https://commons.wikimedia.org/wiki/File:1328101978_Web-page.png
Licence : 

Titre : Official PHP Logo
Auteur : Colin Viebrock
Source : https://commons.wikimedia.org/wiki/File:PHP-logo.svg
Licence : 

Titre : Apache HTTP server logo (2019)
Auteur : The Apache Software Foundation
Source : https://commons.wikimedia.org/w/index.php?title=File:Apache_HTTP_server_logo_(2019-p
Licence : Licence Apache

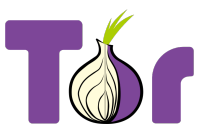
Titre : Nginx logo
Auteur : Igor Sysoev
Source : https://commons.wikimedia.org/w/index.php?title=File:Nginx_logo.svg
Licence : 

Titre : Stack of choc-chip cookies
Auteur : Lisa Fotios
Source : https://commons.wikimedia.org/wiki/File:Cookie_stack.jpg
Licence : 

# Source des images utilisées III

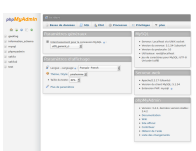
Titre : Third party HTTP cookies
Auteur : Tizio
Source : https://commons.wikimedia.org/wiki/File:Third_party_cookie.png
Licence : CC BY SA

Titre : Facebook Like button
Auteur : Facebook
Source : https://commons.wikimedia.org/wiki/File:Facebook_Like_button.svg
Licence : Public Domain



Titre : Tor logo.
Auteur : The Tor Project, Inc.
Source : https://commons.wikimedia.org/wiki/File:Tor-logo-2011-flat.svg
Licence : CC BY

Titre : logo de MySQL
Auteur : en :MySQL.svg
Source : https://fr.wikipedia.org/w/index.php?title=Fichier:MySQL.svg
Licence : ©



Titre : Screenshot of phpMyAdmin 3.4.2
Auteur : Roccivic
Source : https://commons.wikimedia.org/wiki/File:PhpMyAdmin-main-fr.png

# Source des images utilisées IV

Licence : 

Titre : Exploits of a mom
Auteur : XKCD
Source : `https://xkcd.com/327/`
Licence : ©

# Licence de ce document



**Auteur : Étienne André**

(Source LaTeX disponible aux enseignant·e·s sur demande)

UNIVERSITÉ SORBONNE PARIS NORD

Institut GALILÉE
Université Sorbonne Paris Nord

Version : 29 avril 2024