Programmation Web

Aline Gérard

2018-08-28 Tue

Outline

Initiation à PHP

PHP: Hypertext Preprocessor

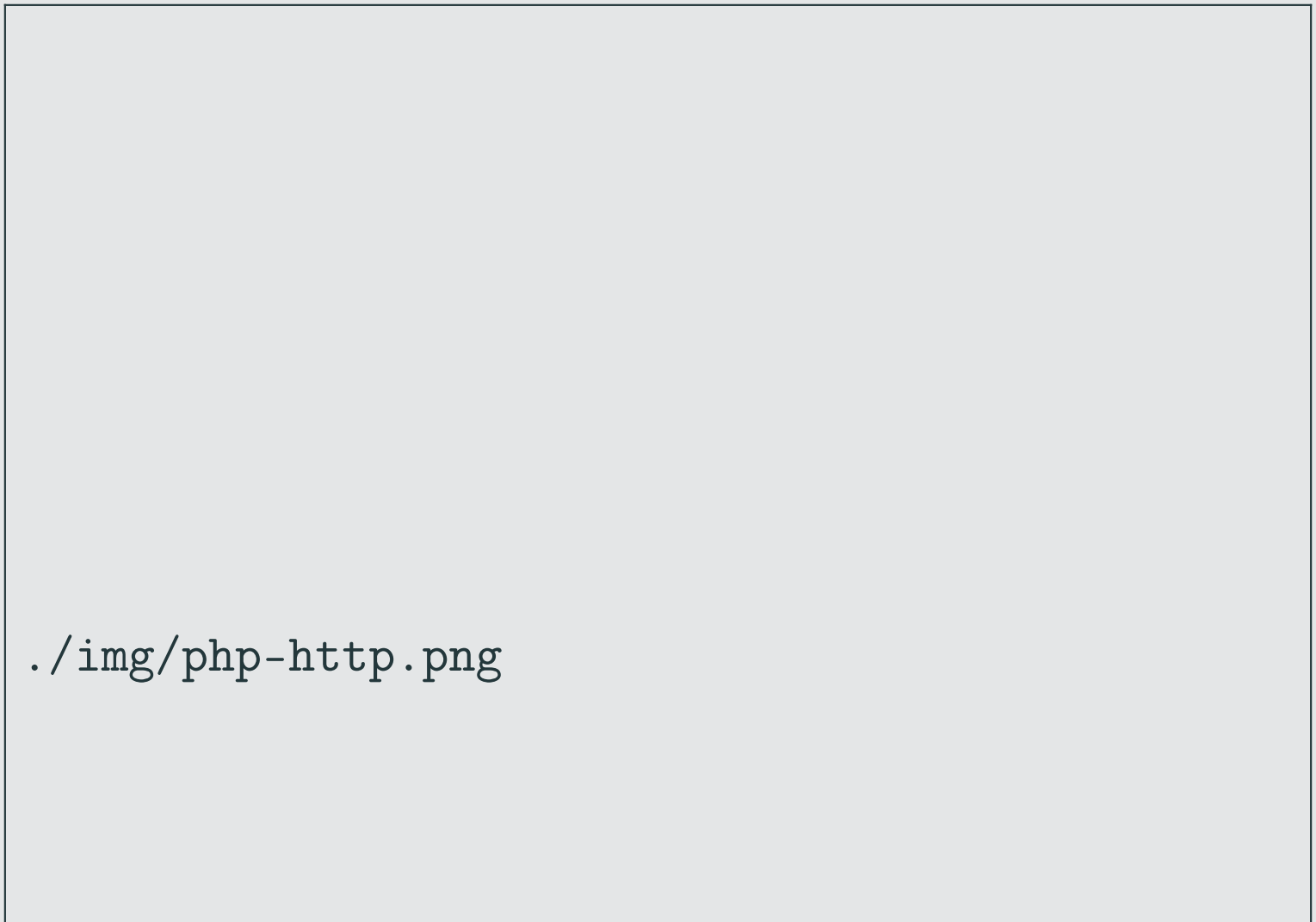
- langage interprété, pas de compilation
- PHP permet la gestion de page web de façon dynamique
- langage souple, de plus en plus objet
- permet de générer des documents HTML, PDF, des images, ...

PHP aujourd'hui

- actuellement PHP 7
- principalement PHP $>$ 5.3 sur les serveurs actuels
- attention, PHP 6 n'existe pas et n'existera jamais !
- utilisation de PHP aujourd'hui

PHP et serveur web

PHP s'utilise conjointement avec un serveur web (Apache, Nginx, ...)



Configurer PHP

se configure via le fichier php.ini

afficher la configuration actuelle

```
<?php
phpinfo();
```

ne pas afficher ces informations en production !

Construction

le code PHP est délimité par

- la balise de début <?php
- et celle de fin ?> qui est optionnelle
- il peut être inséré à l'intérieur d'une page Html
- ou composer un fichier intégralement en PHP.
- les instructions se terminent par ;

Commentaires

```
/*
Ceci est un commentaire
sur plusieurs lignes
*/

// un second commentaire
```

Mon premier script index.php

Hello World

```php
<?php
$msg = "Hello World !";
echo $msg;
```

index.php sera appelé en remplacement de index.html

Du PHP dans mon HTML

```
<?php
// index.php
$msg = "Hello World !";
?>
<!DOCTYPE html>
<html>
    <head>
<title>Mon site web/title>
    </head>
    <body>
<h1><?php echo $msg; ?></h1>
    </body>
</html>
```

PHP n'est pas typé !

- $maVariable = 5;
- $maVariable = "cinq";
- $maVariable = TRUE;
- $monTab = [1, 'un', 2, 'trois', [4, 5]];

Les structures de contrôle

[Toute la doc PHP.net](http://php.net/manual/fr/language.control-structures.php)

if, elseif, else

```php
<?php

if ($expr1) {
  ...
} elseif ($expr2) {
  ...
} else {
  ...
}
```

```php
<?php
if ($expr1) {
  return true;
}
return false;
```

for, foreach

```
<?php
for ($i = 0; $i < 10; $i++) {
  echo $i;
}
<?php
foreach ($tab as $key => $value) {
  echo $key . ' : ' . $value;
}
<?php
foreach ($tab as $value) {
  echo $value;
}
```

Structure de contrôle dans du HTML

```
<h1>Liste des clients</h1>
<ul>
    <?php foreach ($clients as $client): ?>
    <li>
<a href="/show.php?id=<?= $client['id'] ?>">
    <?= $client['name'] ?>
</a>
    </li>
    <?php endforeach ?>
</ul>
```

Structurer avec le Modèle Vue Contrôleur : MVC

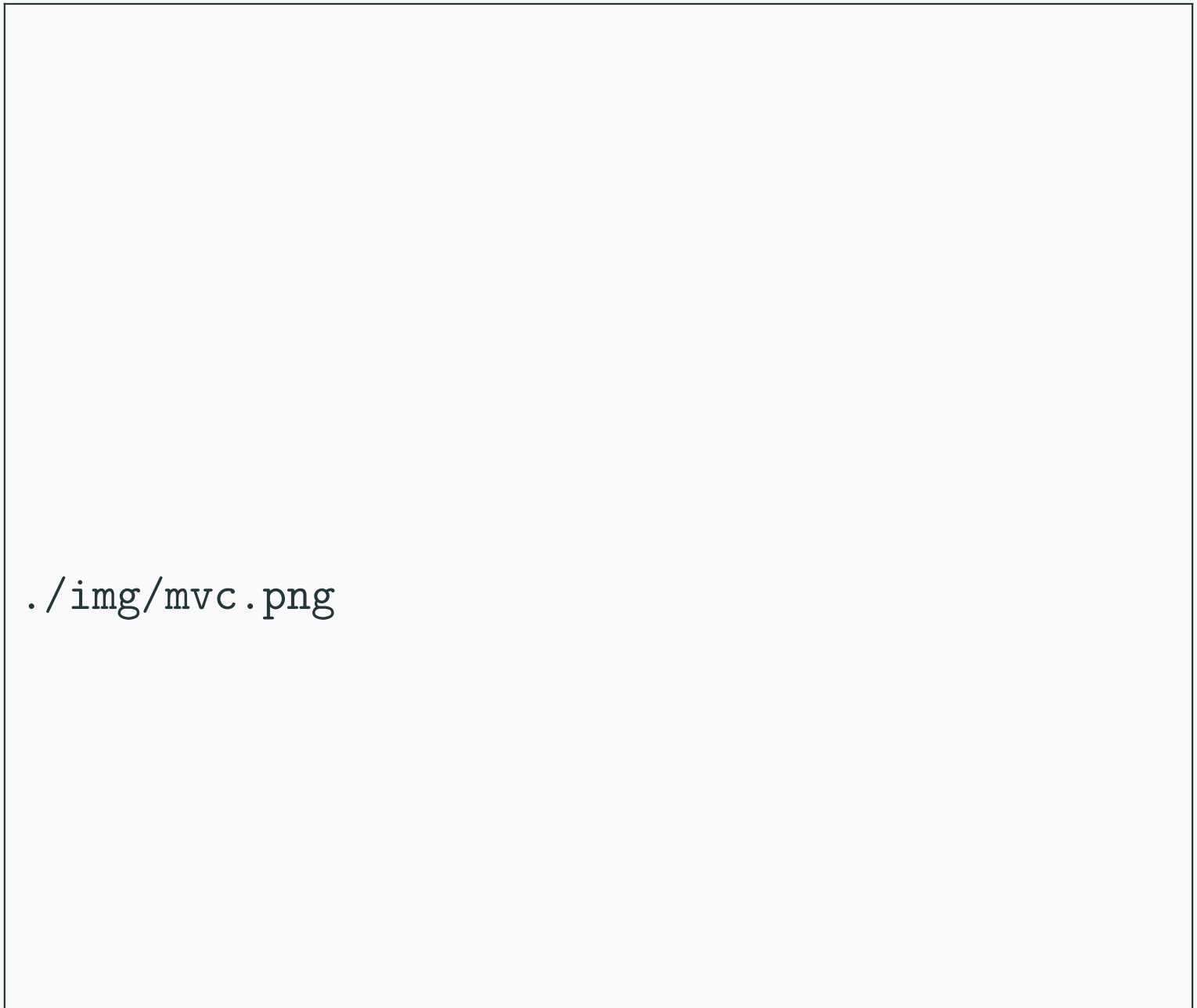
MVC

Modèle traitements et accès aux données. Zéro traitement de l'affichage des données

Vue présentation des données (spus la forme HTML, PDF, ...) à l'aide de template

Contrôleur lien entre l'utilisateur et l'application, fait le lien entre le modèle et la vue.

MVC



Modèle

model.php

```
function get_all_clients() {
  ...
  return $clients;
}
```

ex : templates/client.php

Contrôleur

index.php

```
<?php
require_once 'model.php';

$clients = get_all_clients();

require 'templates/client.php';
```

Décortiquons

- model.php contient des fonctions permettant la récupération de données
- ces données sont affichées dans un template qui constitue la vue
- le template peut-être un fichier PHP contenant du HTML
- le contrôleur qui peut-être index.php récupère les modèles, traite les informations et affiche la vue.
- le découpage permet d'éviter la duplication du code
- et donc une meilleure maintenance de l'application

Les fonctions PHP

Fonction

déclaration et utilisation

```
function test(){ ... }


test();
function test($a, $b = 'toto'){
  ...
}
```

Fontions

fonctions anonymes

```
$test = function(){ ... }

$test();
```

Les inclusions : appel un fichier PHP

include

- include
- include _once // vérifie que le fichier à déjà été appelé

require

- require
- require _once

similaire à include mais stop le programme si le fichier n'existe pas

Découpage des vues

```
<!-- templates/layout.php -->
<!DOCTYPE html>
<html>
    <head>
<title><?= $title ?></title>
    </head>
    <body>
<?= $content ?>
    </body>
</html>
```

templates/client.php

```
<?php $title = 'Liste des clients' ?>

<?php ob_start() ?> <!-- capture le contenu qui va suivre --
    <h1>Liste des clients</h1>
    <ul>
<?php foreach ($clients as $client): ?>
<li>
    <a href="/show.php?id=<?= $client['id'] ?>">
<?= $client['name'] ?>
    </a>
</li>
<?php endforeach ?>
    </ul>
<?php $content = ob_get_clean() ?> <!-- stocke le contenu da
```

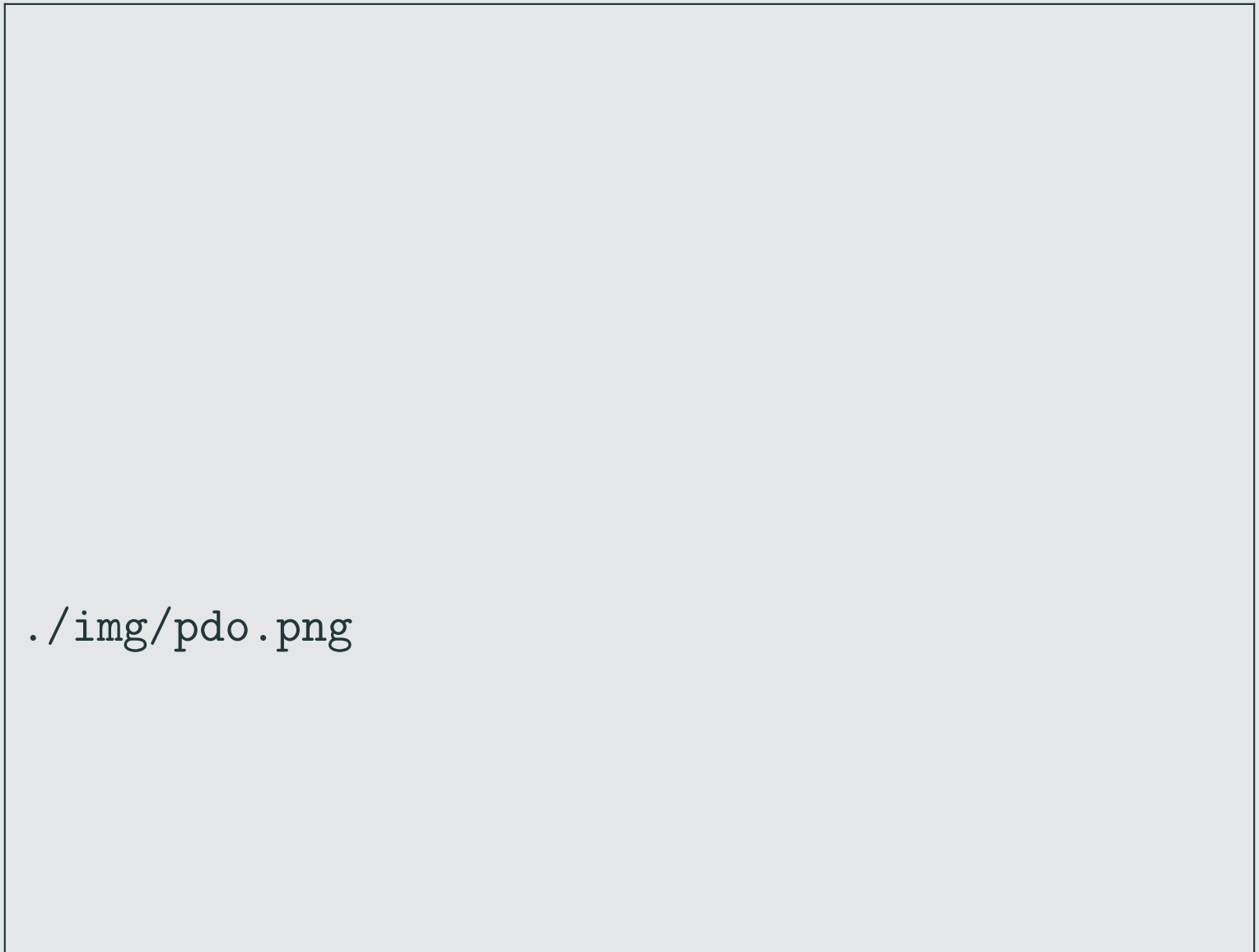
Les formulaires

```
<!-- templates/form.html -->
<!DOCTYPE html>
<html>
    <head>
<title><?= $title ?></title>
    </head>
    <body>
<form action="../form.php" method="POST">
    <input type="text" name="lastname" />
    <input type="text" name="firstname" />
</form>
    </body>
</html>
```

Accès à une base de données

PDO : PHP Data Objects

interface d'accès à de multiples base de données



Utilisation de PDO

```
<?php
$dsn = 'mysql:dbname=testdb;host=127.0.0.1';
$user = 'dbuser';
$password = 'dbpass';

try {
    $dbh = new PDO($dsn, $user, $password);
} catch (PDOException $e) {
    echo 'Connexion échouée : ' . $e->getMessage();
}

$dbh->query();
$dbh->prepare();
$dbh->lastinsertid();
```

Accéder aux attributs

on utilise this

```
<?php
class Client
{

    private $id;

    public function getId(){
    return $this->id;
    }
}
```

Constructeur

le constructeur est appelé lors de la création de l'objet

```
<?php
/* Client.php */
class Client
{

    private $createAt;

public function __construct(){
    $this->createAt = new Date();
}

    public function getClient(){}
}
```

Associer une chaîne de caractère à un objet

```
<?php
/* Client.php */
class Client
{
    private $lastName;
    private $firstName;

    public function __toString(){
return $this->lastName . ' ' . $this->firstName;
    }
}

$client = new Client;
echo $client;
```

Sérialisation

représenter un objet sous forme de texte

- serialize() // transforme un objet en chaîne de caractéres
- unserialize() // redonne sa forme objet

exemple

```
$client = new Client;
var_dump($client);
var_dump(serialize($client));
```

Les espaces de noms

déinition

- permet de définir des classes de mêmes noms
- chacune dans leur propre espace de nom
- utile lors de l'utilisation de librairies
- se déclare en début de fichier avec le mot clé namespace

exemple

```php
<?php
// lib/MonAppli/Client.php
namespace MonAppli;

class Client
{
    ...
}
```

utilisation

```
// index.php
require __DIR__ . '/lib/MonAppli/Client.php;'
$client = new MonAppli\Client();
```

avec l'instruction use

```
// index.php
require __DIR__ . '/lib/MonAppli/Client.php;'
use MonAppli\Client;
$client = new Client();
```

Bonnes pratiques

Chargement automatique : principe

- l'insertion une à une des classes dont on a besoin peut vite devenir laborieux
- l'insertion unique de toutes les classes surcharge inutilement le programme

==> mettre en place l'autochargement (autolaod)

Chargement automatique : en pratique

Créer une fonction d'autoload

```
function loadClass($nameClass)
{
    require 'src/' . $nameClass . '.php';
}

// la fonction loadClass est appelée lors de l'instanciation
spl_autoload_register('loadClass');

$client = new Client();
```

Gestion des erreurs

- activer l'affichage des erreurs
- seulement lors de la phase de développement

dans php.ini :

```
display_errors = On
```

dans un fichier .php :

```
ini_set('display_errors', 1);
echo ini_get('display_errors');
```

Les différents niveaux d'erreurs

E_PARSE	problème de syntaxe (parse error)
E_ERROR	fonctionnalité qui ne peut s'exécuter (fatal error)
E_WARNING	avertissement, le programme s'exécute mais à rencontré une anomalie
E_STRICT	propre à PHP 5, problème de syntaxe
E_NOTICE	relative à la qualité du code
E_DEPRECATED	attention fonctionnalité bientôt dépréciée

Spécifier le niveau d'erreur

dans php.ini

```
; reporter toutes les erreurs
error_reporting = E_ALL
; toutes les erreurs sauf les notices
error_reporting = E_ALL & ~E_NOTICE
```

dans un fichier .php

```
ini_set('display_errors', 1);
error_reporting(E_ALL);
```

Les exceptions : principes

- système permettant l'envoie et la réception d'erreurs
- utilisation de la classe Exception
- stockage d'un code et d'un message d'erreur
- ainsi que le chemin du script et la ligne d'où provient l'erreur

Les exceptions : en pratique

lancer une exception

```
function inTab($test) {
    if ( ! in_array($test, $tab) ){
throw new Exception($test . ' n'est pas contenu dans le teab
    }
}
```

attraper une exception

```
try {
    inTab();
} catch(Exception $e) {
    echo 'L'erreur suivante s'est produite;
    echo $e->getMessage();
}
```

Sécurité : SQL Injection

- envoie de code SQL malveillant via le site web de la cible
- code SQL qui sera exécuté sur le serveur hébergeant le site attaqué
- objectif de l'attaquant :
  - action sur la base de données
  - exécution de code
  - récupérer, détruire, un fichier
  - s'authentifier
  - ...

Sécurité : SQL Injection, exemple

```
$sql = "SELECT * FROM users WHERE id='" .
     $_GET['id'] . "' AND pwd='" .
     $_GET['pwd'] . "'"
     ;
```

Que ce passe-t-il si id vaut :

toto' –

?

Avec PDO on utilisera les requêtes préparées afin de protéger les valeurs

Hash de mot de passe

fct $\text{password}_{\text{hash}}()$ et $\text{password}_{\text{verify}}$

```
<?php
$pwd = password_hash('secret');
if (password_verify($_POST['pwd'], $pwd)) {
    return true;
}
return false;
```

Le format JSON

Javascript Object Notation

format léger pour l'échange de données

```
{
    "nom":"Doe",
    "prénom":"Jane",
    "id":42,
    "ville":"Strasbourg"
}
```

JSON en PHP

```
json_encode();
json_decode();

$client = new Client();
var_dump(json_encode($client));
```

Traitement de fichiers

file_exists() test si le fichier existe

file_get_contents() lit un fichier

file_put_contents() écriture rapide/direct dans un fichier

fopen() ouvre un fichier (ou une url)

fwrite() écriture dans un fichier

fclose() fermeture d'un fichier

Exemple CSV

```
$file = fopen('liste_client', 'r');
while ($client = fgetcsv($file, 1000)) {
    echo $client[0] . "<br />";
}
```

Les sessions : principes

- stockage d'informations sur le serveur
- les données sont enregistrées dans un fichier côté serveur
- les données sont référencées par un identifiant d'une durée courte
- partage des données entre plusieurs fichiers

Les sessions : en pratique

```php
<?php
// création ou récupération d'une session
session_start();

if ( isset($_SESSION['user']) ) {
    echo 'Hello ' . $_SESSION['user'];
} else {
    echo 'Vous n'êtes pas identifié';
}

// destruction du fichier de la session
session_destroy();
// destruction du contenu de la variable de session
unset($_SESSION);
```

Les standards PHP

- PHP-FIG
- défini un standard de programmation
- utilisé par les plus importants framework
- facilite le partage du code
- Tester son code

Ne pas réinventer la roue

- Ressource Packagist
- Awesome PHP
- exemple
    - Traitement de fichiers CSV
    - Mailer