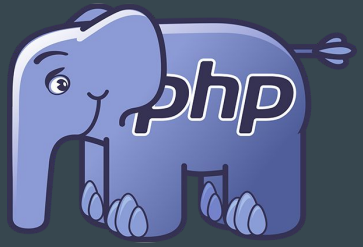
Module de PHP

...

Partie 3 - Utiliser le langage



Le programme de cette partie

- ➔ Découper son code
- ➔ Interagir avec la BDD
  - ◆ mysqli
  - ◆ PDO
- ➔ Récupérer des données de requête (les superglobales)
  - ◆ GET
  - ◆ POST
  - ◆ FILE
  - ◆ SESSION
  - ◆ COOKIE
  - ◆ Les autres
- ➔ Interagir avec le système de fichiers
  - ◆ Les fonctions file_*
  - ◆ Les fonctions f*
  - ◆ Les autres fonctions
- ➔ Échanger des informations
  - ◆ Formater des données
  - ◆ Communiquer

Découper son code

Découper son code

On peut **découper** son code en plusieurs fichiers.

On peut ensuite inclure les fichiers avec `include`.

Interagir avec la BDD

Interagir avec la BDD

On va voir comment interagir avec une BDD :

- ➔ Se connecter
- ➔ Exécuter une requête
- ➔ Récupérer des résultats
- ➔ ...

Aparté sur la Programmation Orientée Objet

La Programmation Orientée Objet en PHP

Les objets en PHP fonctionnent à peu près comme les objets en JavaScript.

```
var objet = new Objet();     // En JS

$objet = new Objet();        // En PHP
```

On accède à l'intérieur d'un objet par la **flèche simple** (->).

```
var p = objet.propriete;       // En JS

$p = $objet->propriete;        // En PHP
```

Nous verrons la Programmation Orientée Objet
plus en détail si il nous reste un peu de temps en fin de module

Fin de l’aparté

Interagir avec la BDD

MySQLi

Connexion

On peut se connecter à une BDD en créant un objet MySQLi.

```
$mysqli = new mysqli('127.0.0.1', '
votre_utilisateur', 'votre_mdp', 'votre_db')
;
```

En cas d'erreur on peut récupérer les variables *connect_errno* et *connect_error* de MySQLi.

Interagir avec la BDD

MySQLi

Les requêtes simples

On peut exécuter une requête en appelant la “fonction” *query* de MySQLi.

```
$result = $mysqli->query("SELECT truc FROM
table LIMIT 10");
```

> En cas d’erreur la fonction renvoie *false*.
> En cas de succès on peut connaître le nombre de lignes renvoyées grâce à l’attribut *num_rows*.

Interagir avec la BDD

MySQLi

Parcourir les résultats

On peut parcourir les résultats grâce à la méthode *fetch_assoc*, qui renvoie un tableau associatif contenant la **prochaine** ligne (ou *false* si pas de prochaine ligne).

En règle général on l'utilise dans une boucle while.

```
while ($actor = $result->fetch_assoc()) {
    // Affiche le nom de l'acteur
    // actuellement parcouru
    echo '<li>' . $actor['first_name'] .
' ' . $actor['last_name'] . </li>';
}
```

Aparté sur les injections SQL

Les injections SQL

Les injections SQL sont des **failles de sécurité** très dangereuses.

Elles arrivent quand on veut utiliser la valeur d'une variable dans une requête : celle-ci pourrait contenir du SQL !

Aussi, la règle d'or c'est : ne pas faire confiance à ce qui vient d'un utilisateur et **échapper** les variables.

```
$bdd->query("SELECT * FROM ma_table WHERE
 nom = $variable");



/**
 * Attention !
 * Il y a là une possible injection SQL !
 */
```

Faites très attention aux injections SQL !

Fin de l’aparté

Interagir avec la BDD

MySQLi

Les requêtes préparées

1) On prépare une requête avec *prepare*.
   (Les “?” marquent les paramètres)
2) On lie les paramètres avec *bind_param*.
3) On exécute avec *execute*.

```php
// On prépare la requête
$stmt = $mysqli->prepare("SELECT quartier
FROM ville WHERE nom = ?");
// On lie les valeurs de nos paramètres
$stmt->bind_param("s", $nom);
// On exécute la requête
$stmt->execute();
```

Interagir avec la BDD

MySQLi

Les requêtes préparées

4) On lie les variables de retour avec *bind_result*.
5) On peut récupérer les résultats !

```php
// On lie les paramètres de retour
$stmt->bind_result($quartier);
// On peut faire ce qu'on veut faire
while ($stmt->fetch()) {
    echo $quartier;
}
// On ferme
$stmt->close();
```

Interagir avec la BDD

PDO

Connexion

On peut se connecter à une BDD en créant un objet PDO. (PDO prend un **DSN** !)

```
$pdo = new PDO('
mysql:host=localhost;dbname=test', 'user', '
passwd');
```

En cas d'erreur, le code lève une **exception**. Il faut donc l'attraper.

Interagir avec la BDD

PDO

Connexion

$e est une **exception** et *getMessage* renvoie le message d'erreur qui lui est associé.

```php
<?php
try {
    $pdo = new PDO('mysql:host=localhost;dbname=test',
 'user', 'passwd');
} catch (PDOException $e) {
    echo 'Erreur : ' . $e->getMessage() . '<br/>';
    die();
}
?>
```

Interagir avec la BDD

PDO

Les requêtes simples

On peut exécuter une requête en appelant la méthode *query* de PDO.

```php
$result = $pdo->query("SELECT truc FROM table LIMIT 10");
```

En cas d'erreur la fonction renvoie *false*.
En cas de succès on peut connaître le nombre de lignes renvoyées grâce à la méthode *rowCount*.

Interagir avec la BDD

PDO

Parcourir les résultats

On peut parcourir les résultats grâce à la méthode *fetch*, qui fonctionne comme le *fetch_assoc* de MySQLi.

En règle générale, on l'utilise dans une boucle while.

```php
while ($row = $result->fetch()) {

    echo $row['truc'];

}
```

Interagir avec la BDD

PDO

Parcourir les résultats

On peut aussi utiliser le résultat de *query* et faire un *foreach* dessus, ça fonctionne pour PDO.

```php
<?php

// Exemple tiré de la documentation officielle

$sql =  'SELECT name, color, calories FROM fruit

ORDER BY name';

foreach  ($conn->query($sql) as $row) {

    print $row['name'] . "\t";

    print $row['color'] . "\t";

    print $row['calories'] . "\n";

}

?>
```

Interagir avec la BDD

PDO

Les requêtes préparées

Ça fonctionne comme pour MySQLi, avec des méthodes p*repare* et *execute*, sauf qu'on peut marquer les paramètres avec "?" ou avec une étiquette ":truc".

```php
<?php
$stmt = $pdo->prepare("SELECT quartier FROM ville WHERE nom
= ?");
// Ou
$stmt = $pdo->prepare("SELECT quartier FROM ville WHERE nom
= :nom");
?>
```

Interagir avec la BDD

PDO

Les requêtes préparées

On peut ensuite binder les paramètres :

1) Avec *bind_param* ou *bind_value*

   Prend le numéro du “?” ou le nom de l’étiquette, la valeur, et le type.

Interagir av
BDD

PDO

Les requêtes préparées

On peut ensuite binder les paramètres :

```php
<?php
$stmt = $pdo->prepare("SELECT quartier FROM ville WHERE nom = ? AND pays = ?");
$stmt->bindParam(1, $nom, PDO::PARAM_STR);  // Le 1er "?"
$stmt->bindValue(2, $pays);                 // Le 2nd "?"
$stmt->execute();

$stmtNom = $pdo->prepare("SELECT quartier FROM ville WHERE nom = :nom AND pays = :pays")
;
$stmtNom->bindParam(':pays', $pays, PDO::PARAM_STR);
$stmtNom->bindValue(':nom', $nom);
$stmtNom->execute();
?>
```

Interagir avec la BDD

PDO

Les requêtes préparées

On peut ensuite binder les paramètres :

2) En passant un array à *execute*
   - normal : les paramètres sont dans l'ordre

     OU
   - associatif : les clefs correspondent aux étiquettes

Interag
B

Les requêtes préparées

On peut ensuite binder les paramètres :

```php
<?php
$stmt = $pdo->prepare("SELECT quartier FROM ville WHERE nom = :nom AND pays = :pays");
$stmt->execute(array(
    'nom' => 'New York',
    'pays' => 'USA'
));

// Ou encore
$stmtNom = $pdo->prepare("SELECT quartier FROM ville WHERE nom = ? AND pays = ?");
$stmtNom->execute(array('New York', 'USA'));
?>
```

Récupérer des données de requête (les superglobales)

Aparté sur les requêtes HTTP

Les requêtes HTTP

HTTP est le protocole qui se cache derrière toute communication web. D'où le "http://" devant les adresses des sites.

Le protocole repose principalement sur des **Headers** qui prennent la forme de paires clefs-valeurs.

Vient ensuite le contenu (*body*) de la requête.

```
HTTP/1.1 200 OK
Date: Sat, 09 Oct 2010 14:28:02 GMT
Accept-Ranges: bytes
Content-Length: 29769
Content-Type: text/html

<!DOCTYPE html... (suivi des 29769 octets
de la page web demandée)
```

Source de l'exemple : developer.mozilla.org

Vous connaissez à présent la face cachée de HTTP !

Aparté sur la confiance utilisateur

La confiance utilisateur

Les données qui proviennent de la requête HTTP proviennent du **client**.

Toutes ces données peuvent être modifiées à volonté par celui-ci.

Comme pour l'injection SQL, il faut **échapper** et **contrôler** la valeur de toutes les variables qui viennent de l'utilisateur.

```
$bdd->query("SELECT * FROM ma_table WHERE
 nom = $variable");



/**
 * Attention !
 * Il y a là une possible injection SQL !
 */
```

Mieux vaut prévenir que guérir !

Fin de l’aparté

Récupérer des données de requête (les superglobales)

GET

On peut récupérer les paramètres d'URL :

http://example.com?param=valeur&truc=3

Côté PHP on récupère une variable **$_GET**, qui est un tableau associatif.

```php
// Dans notre cas
$_GET = array(
    'param' => 'valeur',
    'truc' => 3
);
```

Récupérer des données de requête (les superglobales)

POST

On peut récupérer le contenu des formulaires qui ont la méthode “post”.

Côté PHP on récupère une variable **$_POST**, qui est un tableau associatif.

```php
// Par exemple
$_POST = array(
    'login' => 'YamiShadow',
    'password' => 's€curIté!!'
);
```

Récupérer des données de requête (les superglobales)

FILES

On peut récupérer les fichier envoyés par formulaires.

Côté PHP on récupère une variable **$_FILES**, qui est un tableau associatif, avec comme clef le nom du champ, et en valeur un tableau avec des informations sur le fichier.

Récupérer des données de requête (les superglobales)

FILES

On peut récupérer les fichier envoyés par formulaires.

Côté PHP on récupère une variable **$_FILES**, qui est un

```php
// Par exemple
$_FILES = array(
    'file1' => array(
        'name' => 'MyFile.txt',
        'type' => 'text/plain',
        'tmp_name' => '/tmp/php/php1h4j1o',
        'error' => UPLOAD_ERR_OK,
        'size' => 123
    )
);
```

Récupérer des données de requête (les superglobales)

SESSION

On peut récupérer les données de session.

Côté PHP on récupère une variable **$_SESSION**, qui est un tableau associatif.

Attention, il faut un *session_start*.

Récupérer des données de requête (les superglobales)

SESSION

On peut récupérer les données de session.

```php
session_start();

// Par exemple

$_SESSION = array(

    'pseudo' => 'YamiShadow',

    'is_connected' => true

);
```

C. Récupérer des données de requête (les superglobales)

COOKIE

On peut récupérer les données des cookies de l'utilisateur.

Côté PHP on récupère une variable **$_COOKIE**, qui est un tableau associatif.

```php
// Par exemple
$_COOKIE = array(
    'pseudo' => 'YamiShadow',
    'is_connected' => true
);
```

Récupérer des données de requête (les superglobales)

Les autres

On peut récupérer d'autres variables superglobales.

Côté PHP on a par exemple :

- **$_SERVER** : la configuration serveur.
- **$_REQUEST** : la requête HTTP.
- **$_ENV** : les variables d'environnement

Toutes sont des tableaux associatifs.

Interagir avec le système de fichiers

Interagir avec le système de fichiers

Les fonctions file_*

Il existe les fonctions *file_get_contents et file_put_contents* qui permettent de **récupérer** et d'**insérer** du contenu dans un fichier.

```php
<?php

$contenu = file_get_contents('chemin/fichier.txt');

file_put_contents('chemin/fichier.txt', 'lorem ipsum ... ');

?>
```

Interagir avec le système de fichiers

Les fonctions f*

Il existe plein de fonctions en “f*” pour interagir avec des fichiers :

- *fopen* : ouvrir
- *fclose* : fermer
- *fread* : lire
- *fwrite* : écrire
- *fseek* : placer le curseur

Interagir avec le système de fichiers

Les fonctions f*

```php
<?php

// On ouvre le fichier en lecture et écriture

$file = fopen('chemin/vers/fichier.txt', 'r+');

// On avance de 40 octets

fseek($file, 40);

// On lit 15 octets

$contenu = fread($file, 15);

// On avance à la fin du fichier

fseek($file, filesize('chemin/vers/fichier'));

//  On écrit notre contenu

fwrite($file, $contenu);

// On ferme notre fichier

fclose($file);

?>
```

Interagir avec le système de fichiers

Les autres fonctions

On peut aussi utiliser toutes les fonctions du terminal Linux :

- *mkdir*
- *touch*
- *unlink*
- *chmod*
- ...

Échanger des informations

Échanger des informations

Formater des données

```php
<?php

$json = '{DU JSON ICI}';

$tableau_associatif = json_decode($json, true);

$nouveau_json = json_encode($tableau_associatif);

?>
```

Échanger des informations

Communiquer

En PHP, on peut envoyer un email grâce à la fonction *mail*.

```php
<?php
    $to      = 'personne@example.com';
    $subject = 'le sujet';
    $message = 'Bonjour !';
    $headers = 'From: webmaster@example.com\r\n' .
    'Reply-To: webmaster@example.com\r\n' .
    'X-Mailer: PHP/' . phpversion();

    mail($to, $subject, $message, $headers);
?>
```