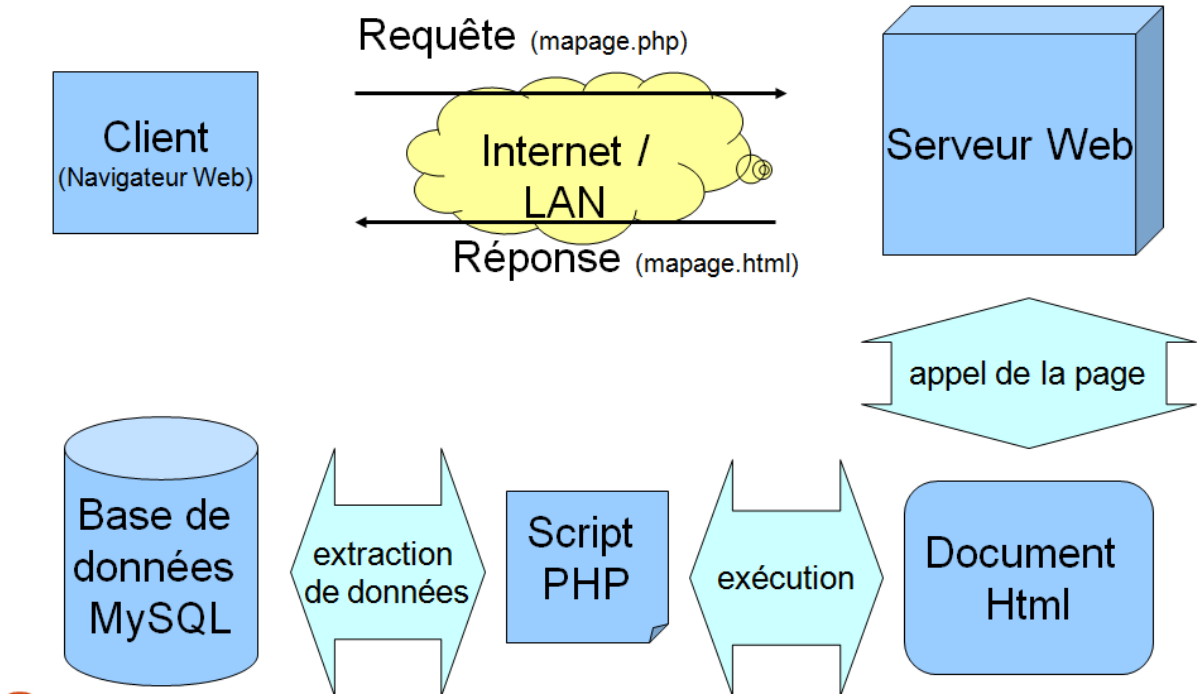
Le langage PHP

1. Présentation

PHP est un langage de script qui s'inclut dans le langage HTML. Le but du langage PHP est de permettre aux développeurs de site web d'écrire rapidement des pages web dynamiques. Ce qui distingue le PHP des langages de script comme le JavaScript est que le code est exécuté sur le serveur et non par le navigateur. Le client ne reçoit que le résultat, sans aucun moyen d'avoir accès au code qui a produit l'affichage. PHP est gratuit, orienté objet, multiplateforme et supporte plusieurs SGBD. Dernière version: PHP 7.2.x.

2. Principe de fonctionnement



3. Environnement de développement

Pour le développement d'un site web dynamique, il faut installer en local PHP, un serveur web et un SGBD. Le serveur web servira pour tester les scripts développés: Apache. Le SGBD servira pour tester la connexion à la base de données et tester les requêtes de manipulation des données de la base: MySQL. Il existe un utilitaire très pratique (EasyPHP ou XAMP) qui installera Apache, PHP, MySQL et phpMyAdmin (une interface gratuite pour la gestion des bases de données MySQL).

PHP && MYSQL

PHP offre 3 APIs différentes pour se connecter à MySQL. Ci-dessous, vous trouverez les APIs fournies par les extensions mysql, mysqli et PDO (PHP Data Objects). Chaque exemple de code crée une connexion à un serveur MySQL s'exécutant sur le domaine "localhost", en utilisant le nom d'utilisateur "root", le mot de passe "password". Et une requête est exécutée pour afficher un étudiant à partir de la table Etudiant.

4. Comparaison des 3 APIs MySQL

```
<?php
// mysqli OO
$mysqli = new mysqli("localhost", "root", "password", "gestion_etudiant");
// Vérification de la connexion
if ($mysqli->connect_errno) {
echo "Echec de la connexion: " . $mysqli->connect_error;
exit();
}
$result = $mysqli->query("SELECT *from Etudiant ");
while ($row = mysqli_fetch_array($result )) {
echo $ row  ['id'].'  '. $row  ['nom'].'  '.$ row  ['prenom'].'<br/>';
}
mysqli_close($mysqli );

// PDO OO
try
{
$pdo = new PDO('mysql:host= localhost;dbname= gestion_etudiant, root, 'password');
}
catch(Exception $e)
{
      die('Erreur : '.$e->getMessage());
}
$result  = $pdo->query("SELECT *from Etudiant ");
while ($row  = $result  ->fetch(PDO::FETCH_ASSOC)) {
   echo $ row  ['id'].'  '. $row  ['nom'].'  '.$ row  ['prenom'].'<br/>';
}

// mysql procedural
$c = mysql_connect("localhost ", " root ", "password") or die(mysql_error());;
mysql_select_db("gestion_etudiant ") or die(mysql_error());
$result = mysql_query("SELECT *from Etudiant ");
while ($row = mysql_fetch_assoc($result))
{
   echo $ row  ['id'].'  '. $row  ['nom'].'  '.$ row  ['prenom'].'<br/>';
}
mysql_close();

?>
```

5. API recommandé

Il est recommandé d'utiliser soit l'extension mysqli, soit l'extension PDO_MySQL. Il n'est pas recommandé d'utiliser l'ancienne extension mysql pour de nouveaux développements sachant qu'elle est obsolète depuis PHP 5.5.0, et sera supprimée dans un futur proche (version 7.0). Une matrice de comparaison détaillant les fonctionnalités est fournie ci-dessous. La performance globale des 3 extensions peut être considérée comme identique. Malgré tout, la performance de l'extension constitue seulement une fraction du temps total d'exécution d'une requête web PHP. Aussi, l'impact est inférieur à 0.1%.

6. Comparaison des fonctionnalités

	mysqli	**PDO_MySQL**	**mysql**
Introduite en PHP version	5.0	5.1	2.0
Inclus avec PHP 5.x	Oui	Oui	Oui
Statut du développement	Active	Active	Uniquement de la maintenance
Cycle de vie	Active	Active	Obsolète
Recommandé pour de nouveaux projets	Oui	Oui	Non
Interface orientée objet	Oui	Oui	Non
Interface procédurale	Oui	Non	Oui
L'API supporte les requêtes non-bloquantes, asynchrones avec mysqlnd	Oui	Non	Non
Connexions persistantes disponibles	Oui	Oui	Oui
L'API supporte les jeux de caractères	Oui	Oui	Oui
L'API supporte les requêtes préparées côté serveur	Oui	Oui	Non
L'API supporte les requêtes préparées côté client	Non	Oui	Non
L'API supporte les procédures stockées	Oui	Oui	Non
L'API supporte les requêtes multiples	Oui	La plupart	Non
L'API supporte les transactions	Oui	Oui	Non
Les transactions peuvent être contrôlées avec SQL	Oui	Oui	Oui
Supporte toutes les fonctionnalités de MySQL 5.1+	Oui	La plupart	Non

7. PDO : Passage de requêtes

Des méthodes permettent de passer des requêtes à l'objet récupéré lors de la connexion.

- PDO::exec(string $statement) : Cette méthode permet de passer et exécuter une requête SQL de type INSERT, UPDATE, DELETE. Elle retourne le nombre de lignes affectées par la requête.

Exemple :

```
$requete = "DELETE FROM etudiant WHERE id = 1";

$resultat = $connexion->exec($requete);

echo $resultat.' suppressions effectuées';
```

- PDO::query(string $statement) : Cette méthode permet de passer et exécuter une requête SQL de type SELECT. Elle retourne le jeu de résultats (s'il y en a) sous forme d'objet PDOStatement.

Exemple :

```
$requete = "SELECT nom, prenom FROM etudiant";

$resultat = $connexion->query($requete);
```

8. PDO : Quelques méthodes de la classe PDOStatement

- PDOStatement::fetch() récupère la ligne suivante d'un jeu de résultats PDO.
- PDOStatement::fetchAll() retourne un tableau contenant toutes les lignes du jeu d'enregistrements PDO.
- PDOStatement::fetchObject() récupère la ligne suivante et la retourne en tant qu'objet.
- PDOStatement::fetchColumn() retourne une colonne depuis la ligne suivante d'un jeu de résultats PDO.
- PDOStatement::rowCount() retourne le nombre de lignes affectées par le dernier appel à la fonction.
- PDOStatement::closeCursor() libère la connexion au serveur, permettant ainsi à d'autres requêtes SQL d'être exécutées. La requête reste dans un état lui permettant d'être de nouveau exécutée. Cette fonction retourne TRUE en cas de succès ou FALSE si une erreur survient.