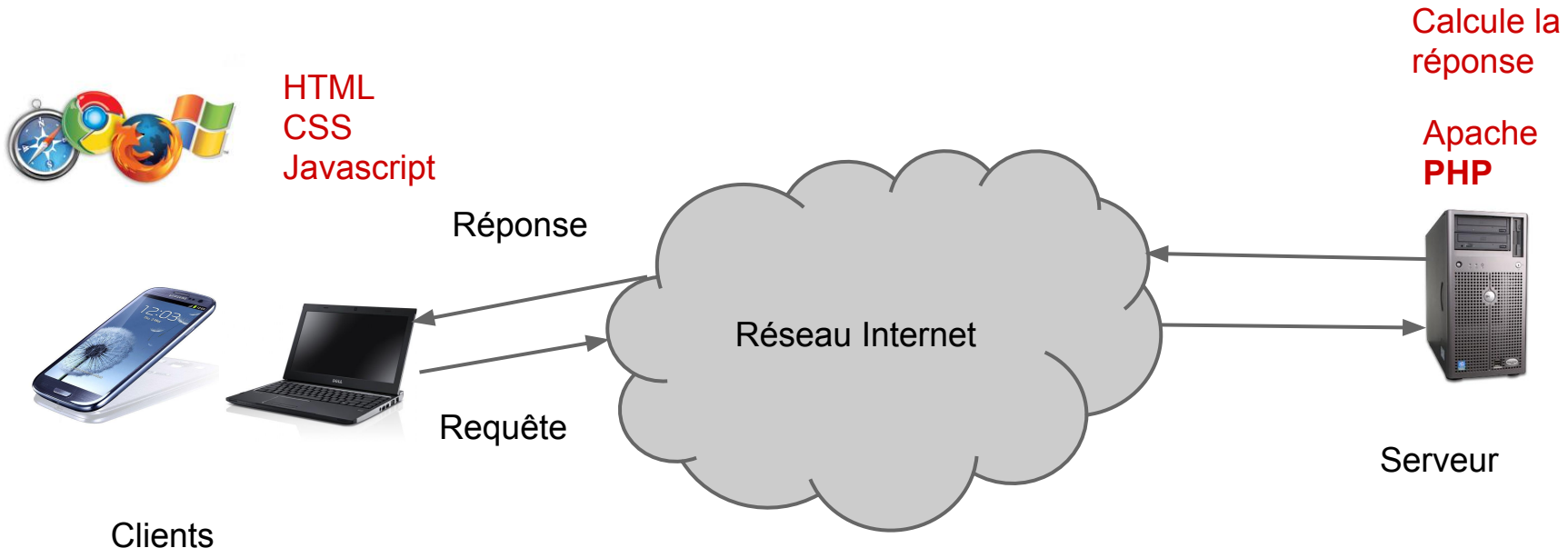
Technologies du web

5 - PHP

Architecture client / serveur



PHP - Généralités

- *Crée en 1994, dérivé du langage Perl*
- *Personal Home Page Tools/Form Interpreter*
- *1997 : PHP 3 PHP Hypertext Preprocessor*
- *2004 : PHP 5*
- ***decembre 2015 : PHP 7***

- Libre sous licence GNU GPL
- http://php.net

PHP

- Langage de script *interprété* côté serveur
  - Pas de phase de compilation
  - Langage dynamique nécessitant un programme pour s'exécuter (l’interpréteur)
- Le plus souvent couplé à un **serveur Apache HTTP** via mod_php
- Adapté à
  - la **génération de HTML**,
  - au **traitement de formulaire**
  - et à l’**accès aux bases de données**

Fichiers PHP

- Extension .php
- Les fichiers contiennent
  - du code HTML
  - du code PHP qui est executé. Le résultat de ce code est intégré au code HTML
- Le fichier PHP est exécuté **à chaque requête**
  - Permet de faire du code dynamique
  - (exemple: renvoyer la date)

Exemple PHP

```
<html>
    <head>
    </head>
    <body>
        <h1>La date du serveur est:</h1>
        <p>
            <?php   echo date("d/m/y")  ?>
        </p>
     </body>
</html>
```

Syntaxe générale

- Entre des balises **<?php ... ?>**
- Commentaires
  - // commentaire sur une ligne
  - /* commentaire sur plusieurs lignes */
  - # autre commentaire

- Chaque instruction se termine par un ;

Fonction echo

- **echo** : la chaîne de caractère sera incluse dans le HTML résultant du script.

```
<?php
   echo '<p>ceci est du html</p>';
   echo date("d/m/y");
?>
```

Les variables

- Le nom commence par $
  - ex: $nom_de_variable, $a, $uneAutreVariable2
- Sensible à la casse

- Pas de déclaration de type, une variable peut contenir un scalaire ou des objets
  - $maVariable = 10;
  - $maVariable = 'toto';

L’assignation

```
$var1 = 10;
$var2 = $var1; // copie par valeur
$var1 = 20;

echo ($var1, $var2);
```

Les types

- Les entiers (integer)

$nombre = 123;

- Les nombres à virgules (doubles)

$a = 123.456;

- Les boolean (boolean)

$a = TRUE; $b = FALSE;

- Les chaînes de caractères (string)

$chaine = "cours de php";

Les type spéciaux

- *null* : absence de valeur
- Les tableaux (*array*)

```
$chiffres = array(6,7,5,3);
$what = array("I","won’t","do","what","you","tell","me");
```

Manipulation des types

```
get_type($var); # retourne le type sous forme de string
is_integer($var);
is_double($var);
is_array($var);

$result = (float)$var; # conversion explicite vers un float
$a = "0" + 2; # conversion implicite
isset($var); #Retourne FALSE si $var n’est pas initialisée ou a la valeur NULL
empty($var); #Retourne TRUE si $var n’est pas initialisée, a la valeur 0, "0", "" ou NULL
```

Les Arrays

- Peut contenir une liste

```
array(6,7,5,3);
```

- Peut contenir un tableau associatif
  - association clé / valeur

```
array("jean"=>20, "vincent"=>22, "julie"=>21);
```

Les Arrays ou tableau associatif

```
$tab1 = array(); $tab2 = array(); // tableaux vides
$tab3 = array("fruit"=>"Pomme", "legume"=>"Haricot"); // avec valeur

// ajout d'élément par index
$tab1[0]= "Roger"; $tab1[1]= "Marcel";
// ajout/maj d'élément par clé
$tab2["papa"] = "Roger"; $tab2["maman"] = "Marcelle";
// accès à un élément
$a = $tab1[1]; $b = $tab2["maman"];
```

Les Arrays suite

```
$tab1[] = 'test3';  # ajoute un élément à la fin
unset($tab1[1]);    # supprime un element
```

Les operations arithmétiques

```
$a = 1;
$b = 2;
$d = $a + 1;
$d = $b − $a;
$d = $a * $b;
$e = 5 / $a ;
$r = $d % 5;
$f = (5 * $a) - ($b * $a + 2);
$a += 2; $b -= 5;
$d++; ++$c ;
```

Les opérations logiques et de comp.

```
$a < $b
$a > $b
$a <= $b
$a >= $b
$a == $b
$a != $b
$a === $b  # comparaison de la valeur ET du type
$a !== $b  # comparaison de la valeur ET du type
($a > $b) && ($a > 0) # ET logique
($a < 0 || !$a) # OU et NON logique
```

Concaténation de chaines

```
$nombre = 2;
$ami = "Roger";
$ami2 = "Norbert";

// concatenation
$moi = "J’ai" . $nombre . " amis. Ce sont " . $ami . " et " .
$ami2 . " . ";

// remplacement DANS la chaîne de caractères
$moi = "J’ai $nombre amis. Ce sont $ami et $ami2.";
```

Condition

```
if (condition) {
   // instructions
} elseif (condition) {
   // instructions
} else {
   // instructions
}
```

Boucles For

```
for($i = 0; $i < 100; $i++) {
   echo $i;
}
```

Boucle sur un array

```
$tab = array("fruit"=>"Pomme", "legume"=>"Haricot");
// Iteration sur les valeurs d'un tableau
foreach($tab as $val) {
   echo $val;
}
// iteration sur clé/valeur d'un tableau
foreach($tab as $cle=>$val) {
   echo $cle . ":" . $val;
}
```

Boucles While

Boucle sur une condition

```
$a = 0;
while($a < 10) {
   echo $a;
   $a = $a + 1;
}
```

Fonctions utiles

count($tableau); #renvoie le nbre d’elements

strlen($chaine); #renvoie le nbre de caractère dans la chaine

explode(",", $chaine); #Transforme une chaine en tableau en separant selon le caractère spécifié (split)

implode(",", $tableau); #operation inverse (join)

trim($chaine); #Renvoie la chaine sans les espaces de debut et de fin

print_r($tableau); #Affiche le contenu d’un tableau

Fonctions utiles (2)

strtolower($chaine); #renvoie la chaine en minuscule

strtoupper($chaine); #renvoie la chaine en majuscule

substr($chaine, $index, $size); # Renvoie la sous chaine commencant à $index et de taille $size

str_replace($old, $new, $chaine); # Remplace $old par $new dans $chaine

strpos($chaine, $souschaine); # Renvoie l'index de $souschaine dans $chaine

Fonctions utiles

sort($tab) : trie le tableau par ses valeurs

array_pop($tab) : récupère et supprime le dernier élément

array_push($tab, $val) : ajoute l'élément à la fin du tableau

array_shift(tab) : récupère et supprime le premier élément

array_unshift(tab, $val) : ajoute l'élément au début du tableau

array_merge($array1, $array2, ...) : fusionne les x tableaux

in_array($needle,$array) : recherche $needle dans les valeurs de $array

array_key_exists($key, $array) : retourne true si $key est une clé de $array

array_flip($array) : inverse les clés et les valeurs

Definition de fonction

```php
<?php
function calcul($a, $b) { // déclaration de fonction
   $resultat = $a + $b ;

   // termine la fonction avec une valeur de retour
   return $resultat;
}
// appel de la fonction
echo("J’ai calculé " . calcul(2, 3) . " et " . calcul(2,2));
?>
```

Fonctions divers...

- Les variables dans la fonction ne sont pas visibles à l'extérieur de celle-ci
- Une fonction n’est déclarée que si la ligne “function” est executée
- La valeur de retour peut être un Array
- On peut donner une valeur par défaut pour les derniers paramètres d’une fonction

```
function foo ($a, $b, $c=10) {
    return $a + $b + $c;
}
foo(1,2,3); // retourne 6
foo(1,2);   // retourne 13
```

Les constantes

Une variable non modifiable.

Sans $

Utilisable dans tout le code

```
define("PHP","PHP Hypertext Preprocessor");
echo PHP;
```

Inclusion de fichiers PHP

Php fournit des fonctions pour inclure des fichiers php dans un autre.

Permet de **réutiliser du code** dans plusieurs fichiers PHP

- **include** : copie et exécute un autre fichier php
- **require** : comme include mais stoppe en cas d’erreur

Include : lib.php

```
<?php

  // definition d’une variable avec du texte
  $str = "<p>ceci est du html</p>";

  function ma_fonction($a, $b) {
      echo  "bonjour " . $a . " et " . $b;
  }
?>
```

main.php

```
<?php
    include('lib.php');  // inclusion d’un autre fichier PHP

    // $str et ma_fonction sont définies dans lib.php
     echo $str;
     $b = ma_function('jean', 'jacques');
    echo '<p>';
    echo $b;
    echo '</p>';
?>
```

Accès aux données

Site avec données

L’objectif d’un site web est de permettre de

- publier une information à un grand nombre d’utilisateurs
- conserver et partager les données envoyées par les utilisateurs (une donnée envoyée est alors consultable par d’autres) : web 2.0

=> Nécessité de conserver et gérer des données

Les données des sites / applis web

- L’information / la donnée est dynamique et change vite.
  - Contenu textuelle (articles, commentaires)
  - Images, Videos etc…

- Elle peut provenir de différentes sources
  - utilisateurs (interne et/ou externe web 2.0)
  - fournisseurs de données, autres sites web…

- Elle peut être utilisée par d’autres (système d’Information d’entreprise, client lourd, autres sites...)

Les données des sites / applis web

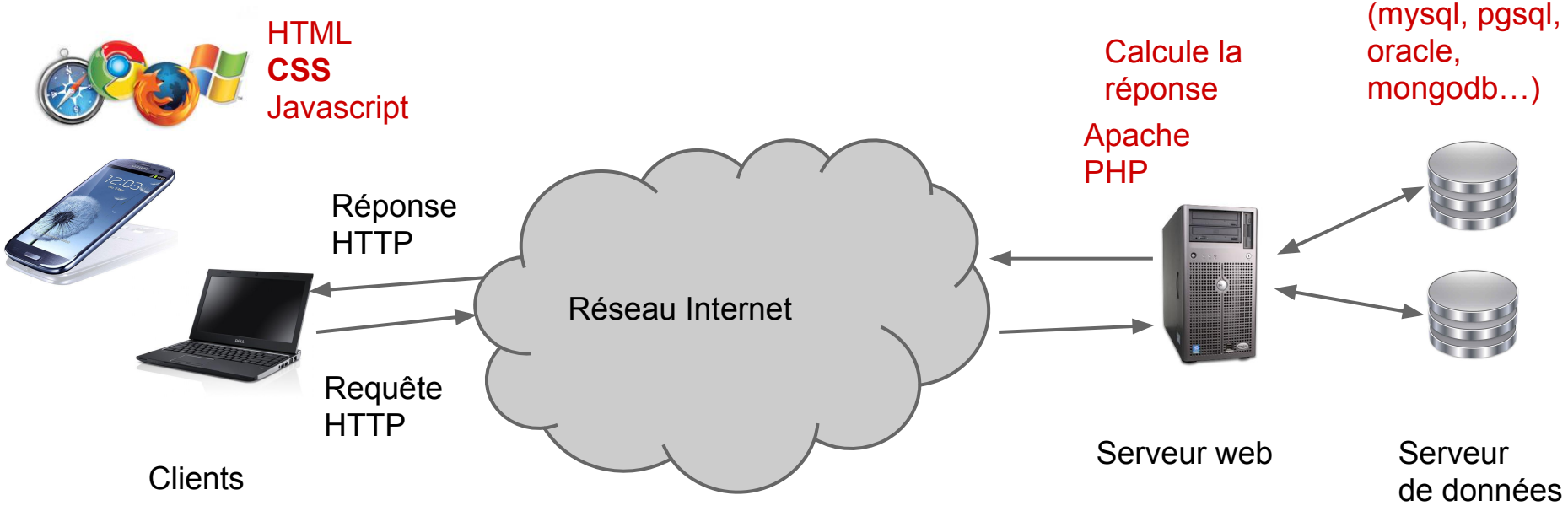
Les données ont leur propre existence.

Les applications web ne sont qu'**une interface de consultation / édition** de ces données.

=> Séparation données / interface web

=> Accès à ces ressources depuis PHP

Architecture 3 tiers



Conserver des données en PHP

Un code PHP est exécuté à chaque requête de manière indépendante : Les variables sont réinitialisées **à chaque fois**.

Comment **conserver des données envoyées**

- **Sessions** : durée de vie limitée de la donnée (ex panier d'achat, authentification…), données **PAR UTILISATEUR et NON PARTAGÉES**

- **Ecriture dans le système de fichier ou dans une base de données** : durée de vie longue (pour des données pérennes), **partage potentiel entre utilisateurs** (ex: commentaires, contenu etc…)

PHP Accès aux données

- Accès aux fichiers
- Persistence d’Array PHP
- Manipulation de format JSON et XML

- Accès à des bases SQL

Système de fichiers

La donnée est stockée directement dans un fichier.

ex : un fichier texte .txt

Facilité d’accès.

Difficile de gérer des données structurées ou de très gros volumes

Lecture/ecriture d’un fichier entier

```
if (file_exists('un_fichier.txt')) {
    $data = file_get_contents('un_fichier.txt');
}

$data2 = 'ceci est du texte',
file_put_contents('un_autre_fichier.txt', $data);
```

Limitations de cette approche

- On ne peut écrire que de la données brute, en générale textuelle

- Si l’on veut de la donnée organisée/structurée, il faut le gérer à la main.

- On ne peut pas gérer de gros volume, car toute la données se retrouve en mémoire ou moment de la lecture du fichier.

Persistance d’une variable (ex array)

On peut sauvegarder le contenu d’une variable dans un fichier.

Transformation de la variable sous la forme d’une chaîne de caractère : la **serialization**.

Avantage : Facile d’emploi

Inconvénient : Non adapté à de gros volumes

Persistance de variables PHP

save.php

```php
$fileName = "data.db";

$data = array("donnée 1", "donnée 2", "donnée 3" ;
// Ecriture de la donnée sur le disk
file_put_contents($fileName, serialize($data))
```

read.php

```php
$fileName = "data.db";
// lit le fichier et recupère le tableau
$data = unserialize(file_get_contents($fileName));

if (!$data) {
  $data = Array();
}
```

Problème de la persistance

- Les données ne sont utilisables que via le programme PHP.

- Pas d’utilisation possible par d’autres programmes.
- Pas de changement ou d’évolution possible du type de donnée stockée

Utilisation de format Semi-structurés

Une alternative au format serialisé (propre à PHP) est d'utiliser un standard pour structurer les données

- XML
- JSON

JSON

Javascript Object Notation

=> un tableau associatif avec une syntaxe compatible avec le langage Javascript

=> type de données supporté :

- chaîne de caractères
- les nombres
- les listes
- les sous tableaux associatifs

Exemple JSON

```
{  "nom" : "bonnet",
   "prenom" : "julien",
   "age" : 33,
   "telephones" : ["0612121212", "0434352324"],
   "adresse" : {
         "rue" : "rue des ecoles",
         "code" : 13000,
         "ville" : "marseille"
   }
}
```

Lire du JSON en PHP

```php
$json = '{"a":1,"b":2,"c":3,"d":4,"e":5}'; // string JSON
 // ou depuis un fichier
 // $json = file_get_contents("mon_fichier.json");

$var = json_decode($json, true);

// $var est false si le json est erronée sinon contient un Array
// a => 1
// b => 2
// etc...
```

Ecrire du JSON en PHP

```php
$a = array('foo' => 'bar','titi' => 'toto');
$json = json_encode($a);

// ecriture dans un fichier
file_put_contents('mon_fichier.json', $json);

// $json contient une chaine de caractère json
// { "foo" : "bar", "titi" : "toto" }
```

Pourquoi JSON

Syntaxe simple

Facile à analyser et à écrire

Lisible par d’autres programmes que votre script PHP.

Utilisable pour transmettre directement via HTTP (web service, ajax)

XML

Langage à balises (comme HTML !!)

Les balises ne sont pas prédéfinies

Plus difficile à parser et à écrire que JSON

Exemple XML

```
<observations>
     <observation>
          <auteur>Jean Paul</auteur>
          <date>18-07-2012</date>
          <taxon>ophrys apifera</taxon>
     </observation>
     <observation>
          <auteur>Jacque Remi</auteur>
          <date>19-07-2012</date>
          <taxon>ophrys fuciflora</taxon>
     </observation>
</observations>
```

Lecture d’un XML

```
$dom = new DomDocument();
$dom->loadXML($chaineXML);


$listeObs = $dom->getElementsByTagName('observation');
foreach($listeObs as $obs) {
    echo $obs->firstChild->nodeValue . "<br />";
}


// voir http://php.net/manual/fr/refs.xml.php
```

Utilisation d’une base de données

Permet de stocker **beaucoup de données**

Avantage : On peut récupérer uniquement les données qui nous intéressent au moment de requête.

Inconvénient : nécessite d’utiliser un moteur de base de données

Types de base de données

Relationnel (SQL)

- **MySQL**
- **SQLite**
- PostgreSQL
- Oracle
- ...

Entrepôt de données (Clés/Valeurs, Documents, Big-Data…)

- Redis
- MongoDB
- Cassandra
- CouchDB

Base de données relationnelle

- Gère des tables
- Les colonnes sont des champs différents
- Les lignes sont les différentes entrées

Les données des tables peuvent être liées entre elles via des relations (1-1, 1-N, N-N)

Table de données : Personnes

Nom	**Prenom**	**Tel**	**Age**
Dupont	Pierre	123	25
Durand	Paul	456	34
Bonnet	Jean	789	45
...	...	...	
...	...	...	
...	...	...	

Base de données MySQL en PHP

Différentes méthodes d’accès à MySQL

- mysql_*
- mysqli_*
- **pdo (non limité à MySQL)**

L’extension doit être activée dans PHP

Connection à la base de données

```php
<?php
try {

    $bdd = new PDO('mysql:host=sql.hebergeur.com;dbname=mabase',
    'pierre.durand', 's3cr3t');
}
catch (Exception $e) {
        die('Erreur : ' . $e->getMessage());
}
?>
```

Les Exceptions

Mécanisme de **gestion d'erreurs**.

Lorsqu'une erreur est détectée, le déroulement du programme s'arrête, et une exception est levée.

Le programme peut définir des procédures de récupération d'erreur en "catchant" (attrapant) les exceptions

Exemple d’Exception

```
try {
// Code à surveiller
}
catch(Exception $except) {
// Gestion des erreurs
    echo ($except->getCode(), $except->getMessage());
    echo ($except->getFile() . ":" . $except->getLine());
}
```

Notion d’objet

PHP supporte les classes et les objets

Un objet est une instanciation d’une classe (le type) … comme en Java ...

```
$monObjet = new NomClasse(...);
```

Notion d’objet

Un objet :

- des données / attributs (type simple ou autres objets)
- des fonctions qui s’appliquent sur ces données (des méthodes)

```
$monObjet->unAttribut;
$monObjet->uneMethode(...);
```

Connexion à la base de données

```php
<?php
try {
    $bdd = new PDO('mysql:host=localhost;dbname=test', 'root', '');

    $bdd = new PDO('mysql:host=sql.hebergeur.com;dbname=mabase',
    'pierre.durand', 's3cr3t');
}
catch (Exception $e) {
        die('Erreur : ' . $e->getMessage());
}
?>
```

Exécuter une requête SQL

```
$query = $bdd->query('SELECT * FROM personnes');

// On parcourt chaque ligne de résultat
// $donnees est un Array avec comme clés les noms des colonnes
while ($donnees = $query->fetch()) {
    echo $donnees['nom'] . ' ' . $donnees['prenom'];
}

$query->closeCursor(); // Termine le traitement de la requête
?>
```

Exemples

Tous les numéros de téléphones des personnes s’appelant “dupont”

```
SELECT tel FROM personnes WHERE nom=’dupont’
```

Toutes les personnes majeures par ordre alphabétique

```
SELECT * FROM personnes WHERE age >= 18 ORDER BY name
```

Paramétrer les requêtes

On peut utiliser des variables dans la requête

```
<?php
$req = $bdd->prepare('SELECT nom, age FROM personnes WHERE nom =
:nom AND age <= :agemax');

$req->execute(array('nom' =>'pierre', 'agemax' => 45));
?>
```

Utiliser des valeurs de formulaires

```
<?php
$req = $bdd->prepare('SELECT nom, age FROM personnes WHERE nom =
:nom AND age <= :agemax');

$req->execute(array('nom' => $_REQUEST['nom'], 'agemax' =>
$_REQUEST['agemax']));
?>
```

Ecrire des données

```
$bdd->exec('INSERT INTO personnes(nom, prenom, tel, age)
VALUES(\'Bernard\', \'Julie\', \'0612121212\', 45)');
```

```
// note on utilise le \ pour échapper le caractère apostrophe
```

Ecrire avec une requête préparée

```
$req = $bdd->prepare('INSERT INTO personnes(nom, prenom, tel, age) VALUES(:nom, :prenom, :tel, :age)');

$req->execute(array(
    'nom' => $nom,
    'prenom' => $prenom,
    'tel' => $tel,
    'age' => $age,
));
```

Modifier

```
$req = $bdd->prepare('UPDATE personnes SET age = :nvage WHERE
nom = :nom');
$req->execute(array(
    'nvage' => $nvage,
    'nom' => $nom
    ));
```

Créer une table

Requête SQL que l'on n'exécute qu'une seule fois.

```
CREATE TABLE Personnes
(
ID int NOT NULL AUTO_INCREMENT,
Nom varchar(255) NOT NULL,
Prenom varchar(255),
Age int,
Tel varchar(255),
PRIMARY KEY (ID)
)
```

Fonctions scalaires et d’aggregation

- Appliquer un traitement aux données en SQL plutôt qu’en PHP

  - UPPER, LOWER, LENGTH
  - COUNT(*), COUNT(DISTINCT *field*), AVG, SUM, MIN, MAX

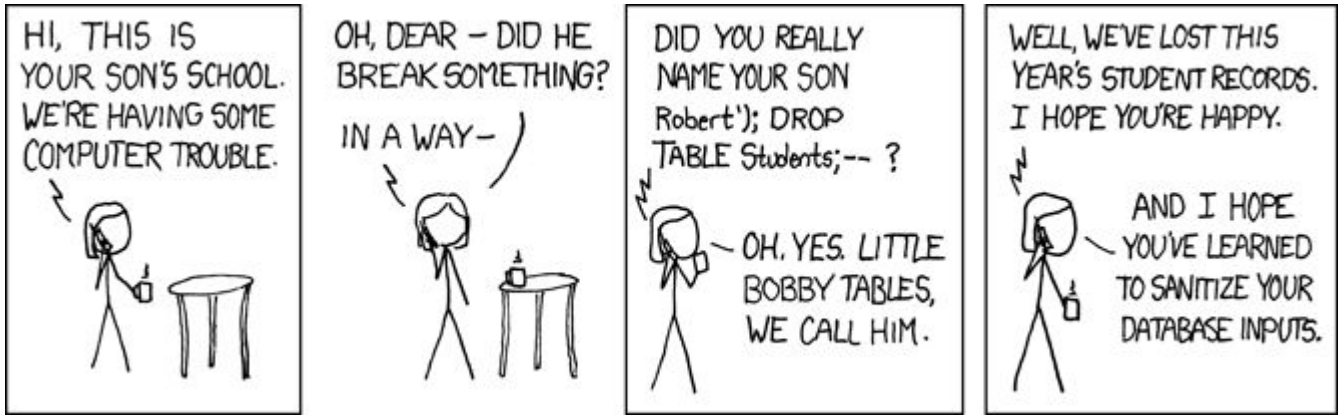
```
SELECT LOWER(nom) AS nom_low FROM personnes
SELECT COUNT(DISTINCT nom) FROM personnes
```

Securité

Injection SQL : en envoyant des valeurs via un formulaire, un utilisateur malveillant peut écrire du code SQL dans un champ.

L’utilisation de requête **préparée**, réduit le risque d’injection de code.

Securité : Injection code



Name: Robert'); DROP TABLE Students; --

Sécurité : Injection HTML/Javascript

Exemple: Ajout d'un commentaire:

Nice site, I think I'll take it. <script>document.location="http://some_attacker/injection.php"</script>

On peut utiliser la fonction ***htmlentities(string)***, pour éviter ce type d'attaque

voir https://en.wikipedia.org/wiki/Code_injection

Conclusion

Une application web interactive a besoin de manipuler des données.

PHP permet entre autre de manipuler des données sous forme de

- Fichiers bruts
- Fichiers formatés en JSON ou XML
- Bases de données relationnelles via SQL

Pour Aller plus loin

http://php.net/manual/fr/book.pdo.php

http://openclassrooms.com/courses/programmez-en-oriente-objet-en-php/manipulation-de-donnees-stockees

empty, isSet, is_null

val	gettype()	empty()	is_null()	isSet()	(bool)
$x = "";	string	true	false	true	false
$x = null;	NULL	true	true	false	false
var $x ; (not set)	NULL	true	true	false	false
$x = array();	Array	true	false	true	false
$x = false;	boolean	true	false	true	false
$x = 15;	integer	false	false	true	true
$x = 1;	integer	false	false	true	true
$x = 0;	integer	true	false	true	false
$x = -1;	integer	false	false	true	true !
$x = "15";	string	false	false	true	true
$x = "1";	string	false	false	true	true
$x = "0";	string	true	false	true	false !
$x = "-1";	string	false	false	true	true
$x = "foo";	string	false	false	true	true
$x = "true";	string	false	false	true	true
$x = "false";	string	false	false	true	true !

Expressions regulières

Mécanisme de recherche à partir d'un motif

```
preg_match ($pattern, $string);
preg_replace ($pattern, $replace, $string);
```

Expressions régulières

```
/PHP/                    # contient PHP
/^PHP/                   # commence par PHP
/PHP$/                   # finit par PHP
/^http:\/\//             # commence par http://
/[PH]+/                  # contient une combinaison de P et de H
/[A-Z][a-z]+/            # contient un mot avec une majuscule
/[0-9]+/                 # contient un nombre entier
/[0-9]+\.?[0-9]*/        # contient un nombre flottant
/P.P/                    # contient P + un caractère + P
/P.*/                    # contient P + 0 ou n caractères
```

Copie par valeur & par référence

```
$var1 = 10;
$var2 = $var1; // copie par valeur
$var3 = &$var1; // copie par reference
// var3 et var1 ont la même case mémoire

$var1 = 20;
echo $var1.' '. $var2.' '.$var3.' '.$var4;
```

Paramètres par référence

```
function foo($var) {
  $var++;
}

function bar(&$var) {
  $var++;
}
$a=5;
foo ($a); // $a vaut toujours 5
bar ($a); // $a vaut 6 maintenant
```

Constructeur d’objet

Une méthode particulière est appelée à la création de l’objet (avec new)

```
function __construct(...)
```

Exemple Personne.class.php

```
class Personne {
    protected $nom;
    public function __construct($nom) {
        $this->nom = $nom;
    }
    public function info() {
        return $this->nom;
    }
}
$moi = new Personne("samuel"); echo($moi->info());
```