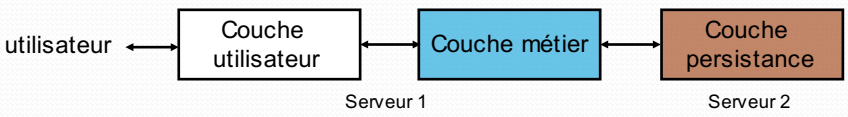
Bases de données

Avec PHP

Couche persistance

Une application web, est généralement basée sur une architecture 3 tiers



Couche persistance

- Comment faire persister ?

- Bases de données
- Bases NoSQL
- Services Web
- Fichiers,
- etc.

Manipulation de BD en PHP

- Ne pas utiliser les mots clés mysql_ (pour éviter attaques type inj. SQL)

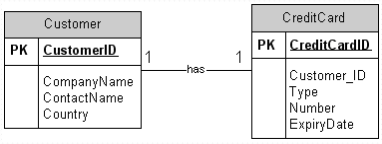
- Utiliser PDO (PHP Data Object)

  - Permet d'éviter les injections SQL
  - Obligatoire en PHP7

2 mots sur les injections SQL

Attaque (bien répandue) de sites web (mal protégés)

exemple:



$custId= $_POST[‘custID’];

$query = "Select * From Customer Where CustomerID = ‘$custID’ ";

Si on tape: cust1' or 1=1 -- => on a la liste de tous les clients

Si on tape:

Cust1'; select * from creditcard => on liste toutes les cartes de crédit

Cust1'; drop table creditcard => on les supprime (tant qu’ à faire)

2 mots sur les injections SQL

Les préventions:

1. Ne pas se connecter sur une base avec des droits admin
2. Valider tout type de paramètre (ne contiennent pas de mot sql, pas de cotes,…
3. Utiliser les PreparedStatement (création de requêtes précompilées)

Les preparedStatement permettent préparer une requête avec des emplacements vide.

Les paramètres externes sont injectés dans les emplacement vides et sont forcement pris comme des chaines

Manipulation de BD en PHP

Exemple d'injection SQL avec PreparedStatement:

Résultat avec cust1' or 1=1 – :

final String sql = "Select * From Customer Where CustomerID = \"cust1' or 1=1 --\""

"cust1' or 1=1 --" est un attribut de la requête pour le *Where*

recherche dans la bd=> ne donnera rien

Manipulation de BD en PHP

PDO

- Connection à BD
- Utilisation de DSN (Data Source name) de la forme *<database>:host=<host>;dbname=<dbname>*
- Exemple : 'mysql:host=localhost;dbname=*nomdevotrebase*';

PDO

- Connection à BD
- Connection avec : *$db = new PDO($dsn, $user, $password);*
- Exception : try {*code*} catch (PDOException $e) {}

PDO

- Préparation d'une requête:
- *$query= 'SELECT FROM foo WHERE id=? AND cat=?';*
  $stmt=?db->prepare($query)
- Ajout des paramètres dans requête:
  $stmt->bindValue(1,120,PDO::PARAM_INT)
  $stmt->bindValue(2,'bar',PDO::PARAM_STR)

Spécifie le type

PDO

A la place du « ? », on peut aussi utiliser *:motclé*

*$query='SELECT * FROM membres where pseudo = :pseudo'*

PDO

- Exécution d'une requête :

  $stmt->execute() // pas de récupération de résultat ici

- Avec récupération de résultat:

```
$stmt->execute();
$results=$stmt->fetchall();
Foreach ($results as $row)
    Print $row['attribut']   // attribut = attribut d'une table
```

- Nb de résultats: *$stmt->rowCount();*

PDO et refactorisation

- Merci de ne pas mettre du Code PDO partout …

- Faire une classe Connection

Exemple:

```
class Connection extends PDO {
private $stmt;
public function __construct($dsn,$username,$password)  {
    parent::__construct($dsn,$username,$password);
    $this->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION); }

    }
```

PDO et refactorisation

```
/** * @param string $query
    * @param array $parameters *
    * @return bool Returns `true` on success, `false` otherwise */
public function executeQuery($query, array $parameters = []) {
        $this->stmt = parent::prepare($query);
        foreach ($parameters as $name => $value) {
            $this->stmt->bindValue($name, $value[0], $value[1]); }
        return $this->stmt->execute();}

public function getResults() { return $this->stmt->fetchall();}
```


PDO et refactorisation

Utilisation:

```
$con=new Connection('mysql:host=localhost;dbname=test', $user, $pass)

$query = 'UPDATE news SET name = :name WHERE id = :id';

$con->executeQuery($query, array(
        ':id' => array($id,PDO::PARAM_STR ),
        ':name' => array($name, ,PDO::PARAM_STR) ));

$con-> getResults();
```

Classe Connection

Connexion à la BD

Méthodes pdo_php de bases

Composée du code en PDO

Rôle:
préparer les requêtes, injecter les paramètres, exécuter la requête

=> Découpler l'accès bas niveau de la BD du reste du projet

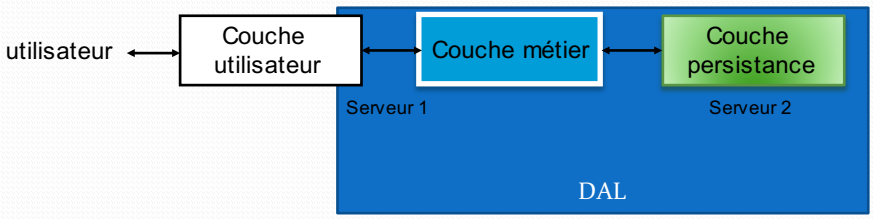
Non rôle:
ne traite pas les erreurs
Pas d'affichage

!! Ne pas retourner d'objet statement (objet retourné par l'appel prepare)
!! Ne pas faire de fetchall (ou autre) or de cette classe
Chacun son rôle

17

Intro aux DAL

- Couche d'accès aux données (DAL)



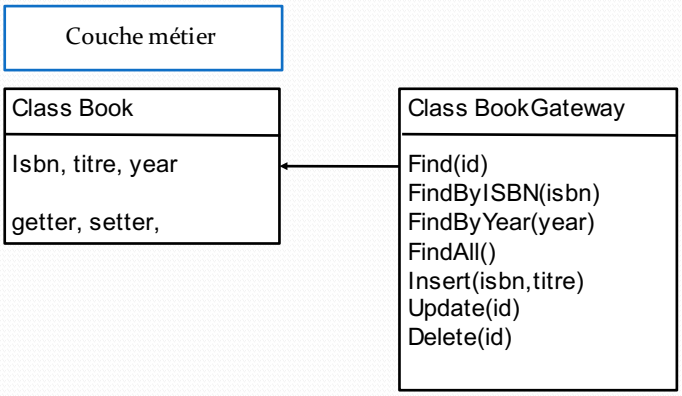
- Couche généralement composée de plusieurs classes
  - Permet de gérer les données à la sauce Objet
- on y trouve des patrons : singleton, factory, voire stratégie
  - Cours suivant

Intro aux DAL

- DAL Data Access Layer
- Plusieurs Modèles :
  - Table Data Gateway
  - Active Record
  - Data Mapper
  - etc.

Intro aux DAL

Table Data Gateway



Intro aux DAL

Classe BookGateway

- Classe qui agit comme passerelle entre tables et objets métier
- Une instance manipule toutes les lignes de la table
- Manipulation type CRUD

```
$b = new BookGateway(new Connection('...'));
// Insert a new record
$id = $b->insert('isbn','titre');

// Update it
$b->update($id,'titre2');
// Delete it
$b->delete($id);
```


Intro aux DAL

```
class BookGateway {
private $con;
        public function __construct(Connection $con) {
        $this->con = $con; }

//méthodes qui font appel à la classe Connection
        public function insert($isbn,$titre) {}
        public function update($id, $titre) {}
        public function delete($id); }
```

Intro aux DAL

Class BookGateway
Exemple méthode insert:

```
public function insert($isbn, $titre) {
$query='INSERT INTO BookTable VALUES (:isbn,:titre)';

$this->$con->executeQuery($query, array( ':isbn' =>
array($isbn,PDO::PARAM_STR),
        ':titre' => array($titre, ,PDO::PARAM_STR) ));

return $this->con->lastInsertId(); }
```

Pour find, findall, il faut extraire les résultats (fetchall, foreach...)