R3.07
Bases de Données - SGBD et PHP

Département Informatique
IUT de Lannion
Université de Rennes

BUT Informatique
4 septembre 2024

Applications Web et BDD

Interface avec PostgreSQL (ou autres SGBDR)

- Rappels d'architecture
- Fonctions d'interfaçage
- Capture des exceptions
- Notions d'utilisateurs (SGBD vs. Web)

Plan du cours

Architecture 3-Tiers

Rappel du fonctionnement d'une application web, vue côté serveur.

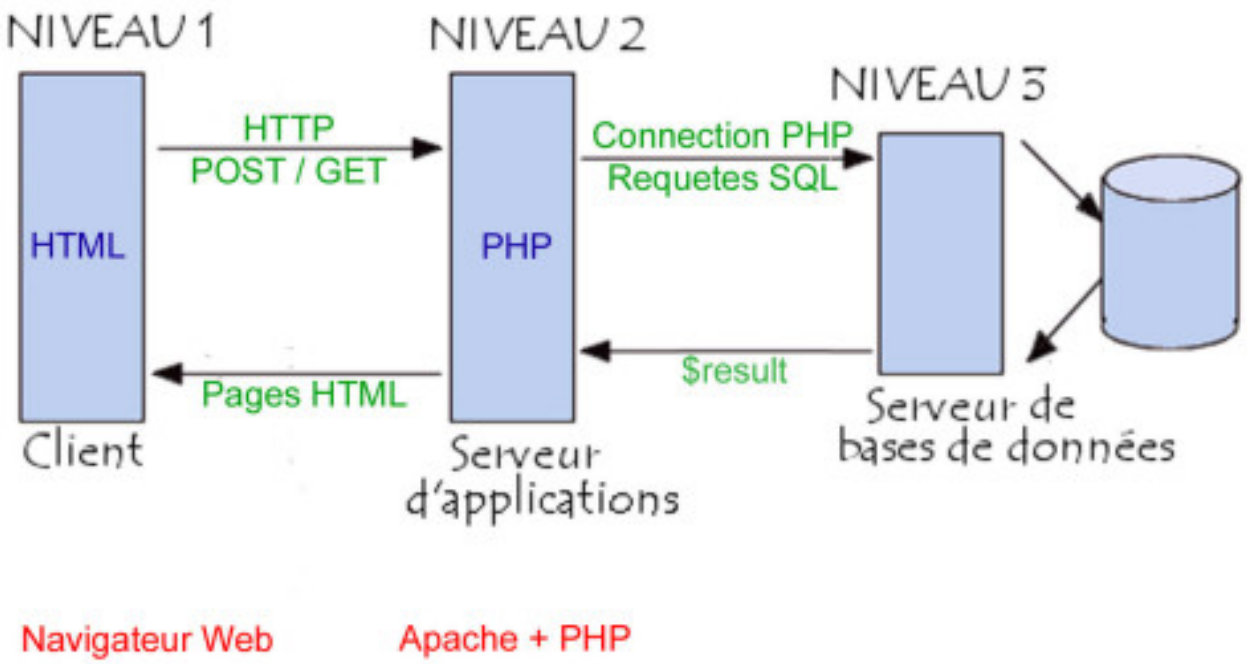


Figure – Architecture 3-tiers - Client-Application-SGBD

Décomposition en MVC

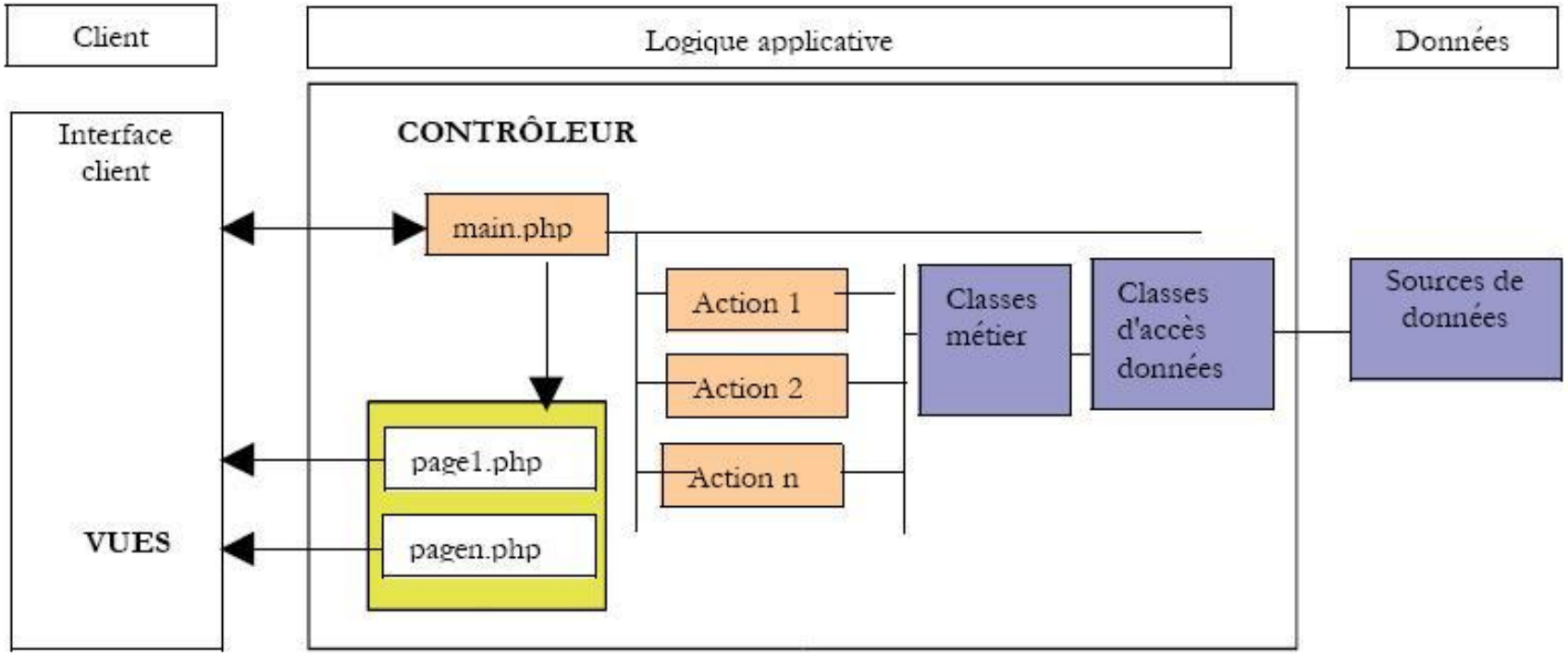
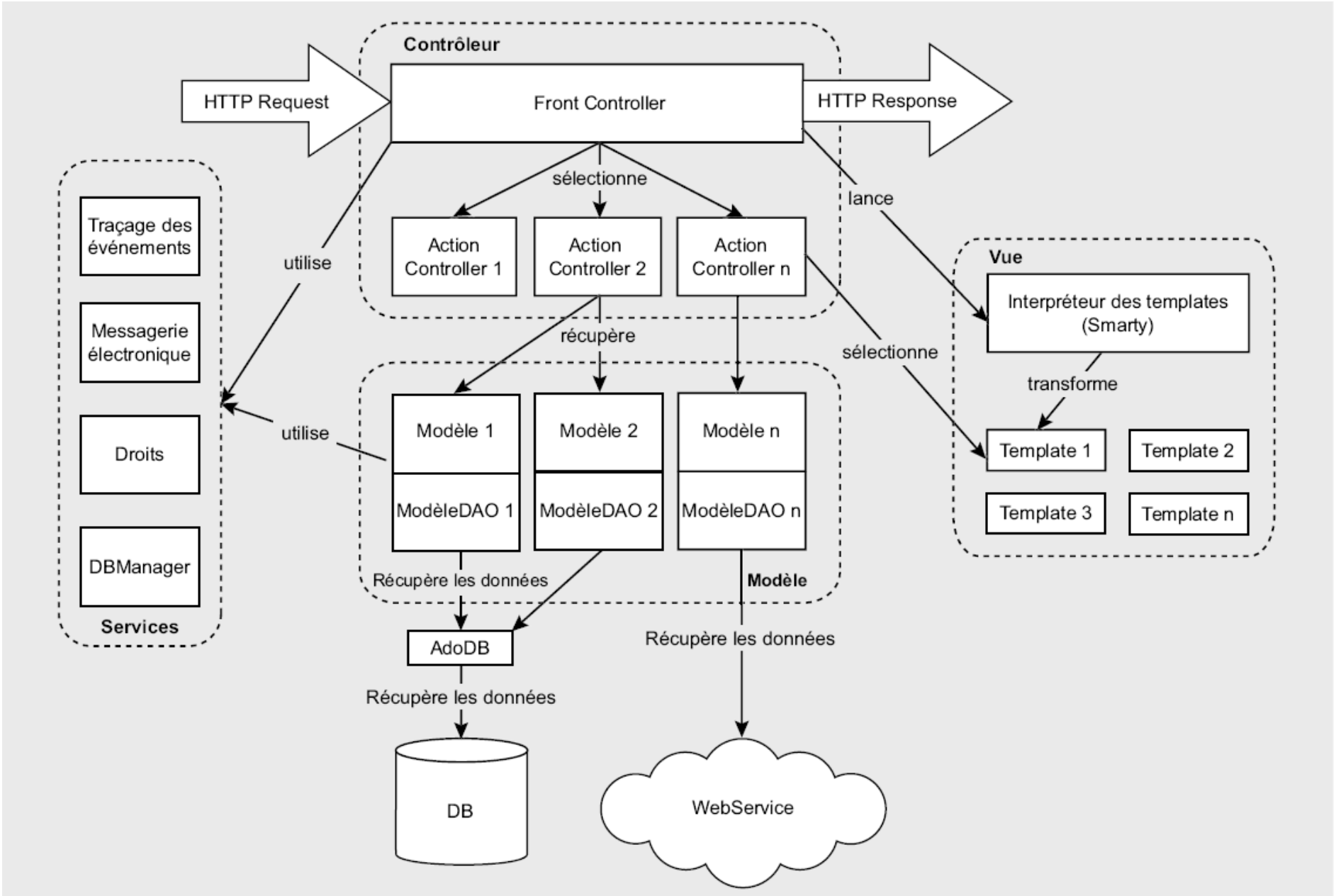


Figure – Model-View-Controller

MVC - une vue plus large



Interfaçage avec les SGBD

Un exemple pour Oracle : Utilisation de Oracle Call Interface (OCI)

```
resource ocilogon (string username,
                   string password
                   [, string db
                   [, string charset]])

resource ociparse(resource conn, string query)

bool ociexecute(resource stmt [, int mode])
```

Interfaçage avec les SGBD

```php
<?php
  $conn = ocilogon("username","password");
  $query = "SELECT nom_arbre FROM arbre";
  $statement = OCIParse ($conn, $query);
  OCIExecute ($statement);
  while (OCIFetchInto ($statement,
                       $row, OCI_ASSOC)){
    print $row['nom_arbre'];
  }
?>
```

Bilan :

+ L'interface est optimisée pour le SGBD
- Dans le cadre d'une migration, tout le code est à revoir

=> Utiliser une couche logicielle d'abstraction

Interfaçage avec les SGBD - PDO

Interfaçage via une couche d'abstraction

- ADODB depuis PHP 4.1
- PDO (PHP Data Object) depuis PHP 5

Intérêt :

- Ne plus se soucier du SGBD avec lequel on travaille
- Plus de portabilité
- Séparation plus nette des couches logicielles (encore que ...)

Interfaçage avec les SGBD - PDO principes

Source incontournable : php.net

Etapes d'une interaction avec le SGBD

- Connexion
- Préparation d'une requête
- Exécution de la requête
- Récupération des résultats
- Fermeture de la connexion

Interfaçage avec les SGBD - PDO Connexion

Comment se connecter :

- Créer des instances de la classe de base de PDO
- Peu importe le SGBD, on utilise la classe PDO
- Fournir des paramètres pour spécifier la source de la base de données (connue en tant que DSN)
- Fournir optionnellement le nom d'utilisateur et le mot de passe (s'il y en a un)

```php
<?php
$dbh = new PDO(
       'postgres:host=servbdd;dbname=pg_test',
       $user, $pass);
?>
```

Erreurs de connexion : l'objet n'est pas instancié (et il est vide).

Interfaçage avec les SGBD - PDO Déconnexion

La connexion est active tant que l'objet PDO l'est aussi.

```php
$dbh = new PDO(
    'postgres:host=servbdd;dbname=pg_test',
    $user, $pass);
// utiliser la connexion ici
$sth = $dbh->one_pdo_method(...);
// et maintenant, fermez la connexion
$sth = null;
$dbh = null;
```

Si on ne cloture pas la connexion explicitement, PHP le fera en fin de script.

Interfaçage avec les SGBD - Connexion persistante

- Connexions pas fermées en fin de script, et mises en cache
- Réutilisation dans d'autres scripts si mêmes paramètres
- Conséquence : une application Web plus rapide

```php
$dbh = new PDO(
    'postgres:host=servbdd;dbname=pg_test',
    $user, $pass,
    array(PDO::ATTR_PERSISTENT => true)
);
```

Remarque : l'utilisation de la valeur `PDO::ATTR_PERSISTENT` doit être faite au moment de l'instanciation de l'objet PDO pour être effective.

Interfaçage avec les SGBD - Lecture

Constantes prédéfinies :

- pour orienter le comportement des méthodes de la classe PDO
- exemples :
  - format des données récupérées : tableau associatif ou à index numérique
  - valeur de l'autocommit
  - gestion du mode d'erreur
  - persistance de la connexion

```
$connection->setAttribute(
  PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_OBJ);
```

Interfaçage avec les SGBD - Lecture

Méthode `PDO::query()` :

```php
$dbh = new PDO(
    'postgres:host=servbdd;dbname=pg_test',
    $user, $pass);
$sql =  'SELECT name, color, calories FROM fruit';
foreach  ($conn->query($sql) as $row) {
    print $row['name'] . "\t";
    print  $row['color'] . "\t";
    print $row['calories'] . "\n";
}
```

donnera

```
apple   red     150
banana  yellow  250
kiwi    brown   75
...
```

Interfaçage avec les SGBD - Lecture

Methode `PDO::fetchAll()`

```php
$dbh = new PDO(
    'postgres:host=servbdd;dbname=pg_test',
    $user, $pass);
$sth = $dbh->query('SELECT * FROM countries');
$rows = $stm->fetchAll();

foreach($rows as $row) {
  print("$row[0] $row[1] $row[2]\n");
  print("$row['id'] $row['name'] ".
        "$row['population']\n");
}
// Difference entre les deux affichages ?
```

Interfaçage avec les SGBD - Modification de données

Ces requêtes ne renvoient *a priori* pas de données, juste un résultat de succès.

```
$stmt = $dbh->prepare(
  "INSERT INTO REGISTRY (name, value)
   VALUES ('$name', '$value')"
);
$stmt->execute();
$dbh = null;
```

- Le résultat de la méthode `execute()` est un booléen
- Une exception est lancée dans les versions récentes de PHP

Interfaçage avec les SGBD - Requêtes préparées

Qu'est-ce qu'une requête préparée

- sorte de modèle compilé pour le SQL que vous voulez exécuter, qui peut être personnalisé
- offre deux fonctionnalités essentielles
  - La requête ne doit être analysée (ou préparée) qu'une seule fois, mais peut être exécutée plusieurs fois avec des paramètres identiques ou différents.
  - Les paramètres pour préparer les requêtes n'ont pas besoin d'être entre guillemets ; aucune injection SQL n'est possible

Interfaçage avec les SGBD - Requêtes préparées

Exemple tiré de la documentation d'insertions répetées à l'aide de marqueurs nommés.

```php
$stmt = $dbh->prepare(
    "INSERT INTO REGISTRY (name, value)
     VALUES (:name, :value)");
$stmt->bindParam(':name', $name);
$stmt->bindParam(':value', $value);

// insertion d'une ligne
$name = 'one';
$value = 1;
$stmt->execute();

// une autre ligne avec d'autres valeurs
$name = 'two';
$value = 2;
$stmt->execute();
```

Interfaçage avec les SGBD - Procédures stockées

Si le pilote de la base de données le prend en charge, vous pouvez également lier des paramètres aussi bien pour l'entrée que pour la sortie.

```php
$stmt = $dbh->prepare(
  "CALL takes_string_returns_string(:in_out_param)");
$stmt->bindParam(
  ':in_out_param', $value,
  PDO::PARAM_STR|PDO::PARAM_INPUT_OUTPUT,
  4000); // longueur prevue

// Appel de la procedure stockee
$value = 'hello';
$stmt->execute();
echo "La procedure retourne : $value\n";
```

Gestion des erreurs - Capture des exceptions

- En cas d'erreur, un objet `PDOException` est "lancé"
- Deux gestions :
  - Attraper cette exception
  - Laisser le gestionnaire global d'exception la traiter (défini via la fonction `set_exception_handler()`)

```php
$pdo = new PDO (whatever);
$pdo->setAttribute(PDO::ATTR_ERRMODE,
                   PDO::ERRMODE_EXCEPTION);
try {
  $pdo->exec ("QUERY WITH SYNTAX ERROR");
} catch (PDOException $e) {
  if ($e->getCode() == '2A000')
    echo "Syntax Error: ".$e->getMessage();
}
```

Gestion des erreurs - Codes d'erreurs de PostgreSQL

On peut trouver dans la documentation PostgreSQL les différents codes d'erreurs PostgreSQL Error Codes et réagir en conséquence.

Quelques exemples :

- `08004 : sqlserver_rejected_establishment_of_sqlconnection`
- `20000 : data_exception`
- `20007 : invalid_datetime_format`
- `23000 : integrity_constraint_violation`
- `23502 : not_null_violation`

Les erreurs sont rangées par classe, dont le numéro est indiqué sur les deux premiers digits du code.

Gestion des erreurs - Exemple

Tiré de la documentation PHP, le code

```php
// Provoque une erreur
// si la table BONES n'existe pas
$dbh->exec(
  "INSERT INTO bones(skull) VALUES ('lucy')"
);
echo "\nPDO::errorCode(): ", $dbh->errorCode();
```

affiche le résultat suivant :

```
PDO::errorCode(): 42S02
```

Gestion des erreurs - Transactions

```php
try {
  $dbh->setAttribute(PDO::ATTR_ERRMODE,
                     PDO::ERRMODE_EXCEPTION);

  $dbh->beginTransaction();
  $dbh->exec("insert into staff (id, first, last)
                values (23, 'Joe', 'Bloggs')");
  $dbh->exec("insert into
      salarychange (id, amount, changedate)
      values (23, 50000, NOW())");
  $dbh->commit();

} catch (Exception $e) {
  $dbh->rollBack();
  echo "Failed: " . $e->getMessage();
}
```

Notions d'utilisateurs (SGBD vs. Web)

Discussion à propos des utilisateurs :

- utilisateur de l'applicatif
- vs. utilisateur du SGBD

Qui utilise la base ?

- un utilisateur défini pour que les scripts accèdent à la base
- les moyens d'accès à la base (login, mot de passe, base) sont définis au niveau des scripts PHP

Envisager la définition de types d'utilisateurs pour l'applicatif (profils)

- gestion interne à l'applicatif, ou
- délégation partielle au SGBD en créant autant de comptes que de profils