

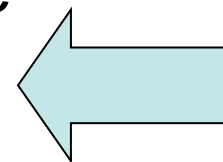
Module MI3

PHP et architecture MVC
pour les applications web

2016-2015

Plan du cours

1. Rappels sur la Programmation Web
2. Présentation de PHP
3. Vers une plus grande Structuration du Code
- 4. PHP et l'accès aux Bases de Données**
5. Principe d'une architecture MVC dans les applications web



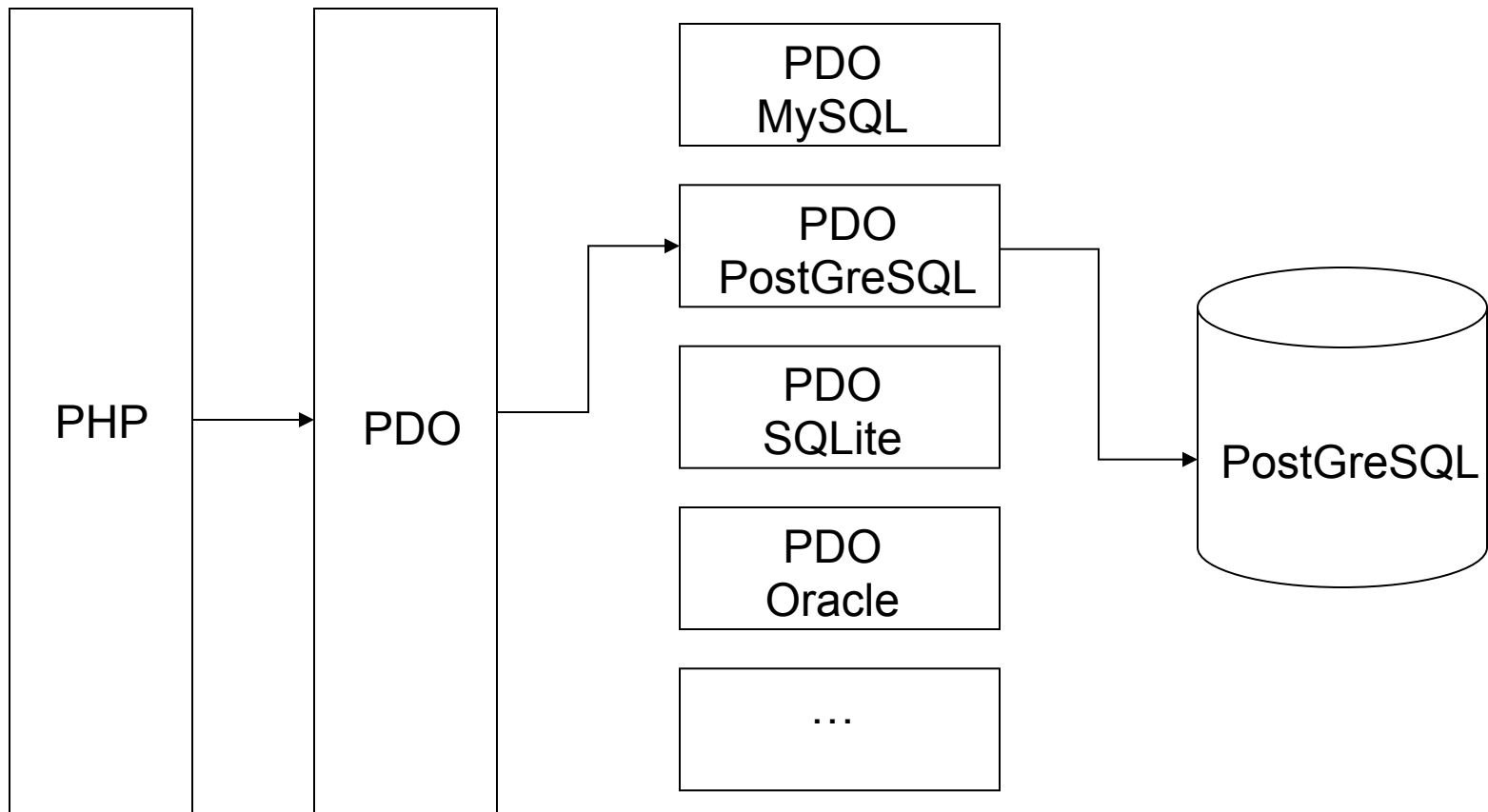
Module MI3

PHP et l'accès aux bases de données

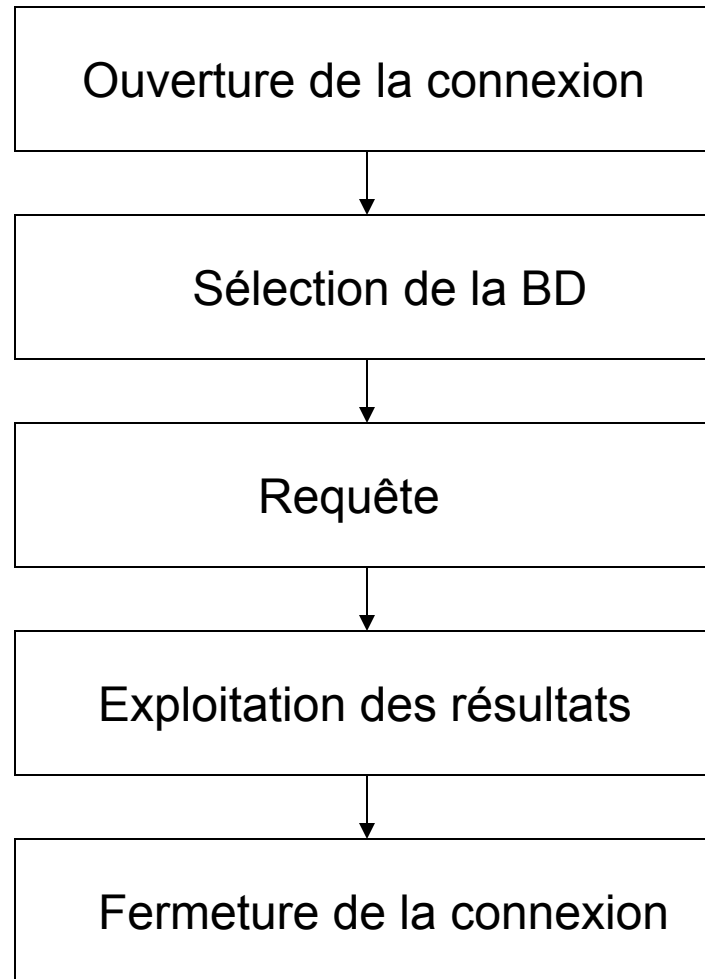
Accès aux Bases de Données

- Une des grandes forces de PHP => le support de nombreuses bases de données:
 - PostgreSQL, Ingres, Oracle, Sybase, IBM DB2, MySQL, Informix,...
 - PHP 5 :
 - intègre un SGBDR, SQLite3
 - Interface PDO
- PDO (PHP Data Object) => interface pour accéder à une base de données depuis PHP
 - approche objet
 - socle commun pour les connecteurs vers les SGBD,
 - plus rapide que d'autres systèmes d'abstraction (PEAR DB, AdoDB,...)

Architecture des drivers PDO



Etapes d'accès à une base de données



3 classes principales

- Classe PDO: correspond au lien avec la BD
- Classe PDOException: permet le traitement des erreurs
- Classe PDOStatement : correspond aux requêtes et aux résultats

Ouverture de connexion

- Création d'une instance de la classe PDO:
 - Le paramètre du constructeur de la classe est le DSN : *Data Source Name*

```
$dbh = new PDO(DSN)
```

- DSN pour MySQL

```
$dbh = new PDO('mysql:host=localhost;  
dbname=basetest', $user, $pass);
```

- DSN pour PostgreSQL

```
$dbh = new PDO('pgsql:host=localhost  
dbname=basetest', $user, $pass);
```


Gestion des erreurs, fermeture de connexion

- Gestion des erreurs

```
try {  
    $dbh = new PDO('pgsql:host=localhost  
    dbname=basetest', $user, $pass ');  
}  
catch (PDOException $e) {  
    die("erreur de connexion:".$e->getMessage());  
}
```

- Fermeture (explicitement ou implicitement à la fin du script)

```
if ($dbh) {  
    $dbh=NULL;  
}
```

Effectuer une requête

2 méthodes de l'objet PDO : `exec()` et `query()`

- `exec()`
 - pour les requêtes qui ne renvoient pas de résultats (UPDATE, INSERT, DELETE)
 - la méthode renvoie le nb de lignes traitées
- `query()`
 - pour les requêtes qui renvoient un résultat (SELECT)
 - la méthode renvoie une instance de l'objet PDOStatement

Requête de Sélection

- La méthode `query()` renvoie une instance de l'objet `PDOStatement`

```
$req="select * from etape";  
$sth=$dbh->query($req);
```

- 2 méthodes pour manipuler les données renvoyées:

`fetchAll()` : retourne l'ensemble des données

`fetch()` : permet une lecture séquentielle du résultat

- Choisir le format des résultats (3 possibilités):

```
$result=$sth->fetchAll(PDO::FETCH_ASSOC);  
$result=$sth->fetchAll(PDO::FETCH_CLASS);  
$result=$sth->fetchAll(PDO::FETCH_BOTH);
```

Requête de Sélection

- `$result=$sth>fetchAll(PDO::FETCH_ASSOC);`
Retourne un tableau associatif indexé par les noms de colonnes
- `$result=$sth->fetchAll(PDO::FETCH_BOTH);`
Retourne un tableau associatif indexé par les noms de colonnes, mais aussi par les numéros des colonnes (*mode par défaut*)
- `$result=$sth->fetchAll(PDO::FETCH_CLASS, "MaClasse");`
Retourne un tableau d'objets dont les propriétés correspondent aux noms des colonnes
- D'autres modes...

Exemple avec PDO::FETCH_ASSOC

```
$req="SELECT nc,ne,nomv FROM etape";
```

```
$sth=$dbh->query($req);
```

```
$result=$sth->fetchAll(PDO::FETCH_ASSOC);
```

```
//affichage de tout le tableau
```

```
foreach ($result as $row){  
    echo $row['nc']; echo '-';  
    echo $row['ne']; echo '-';  
    echo $row['nomv']; echo '<br />';  
}
```

```
//affichage colonne nomv de la 1ère ligne
```

```
echo $result[0]['nomv'];
```

Exemple avec PDO::FETCH_OBJ

```
$req="SELECT nc,ne,nomv FROM etape";
```

```
$sth=$dbh->query($req);
```

```
$result=$sth->fetchAll(PDO::FETCH_OBJ);
```

```
//affichage nomv de la 2ème ligne
```

```
$obj=$result[1];
```

```
echo $obj->nomv;
```

Echappement

Problème: SQL a ses propres caractères spéciaux et délimiteurs

- *Exemple:*

```
$nom="O'Sullivan";
```

```
$nom=$dbh->quote ($nom) ;
```

```
$sql="INSERT INTO ville (nomv,pays,descripville)  
values ($nom,'Irlande','belle ville!');"
```

```
$dbh->exec ($sql) ;
```

```
$sql="DELETE from ville where nomv=$nom";
```

```
$dbh->exec ($sql) ;
```

Requêtes "préparées"

Pour plus de sécurité => requêtes "préparées":

- **Principe:**

- Créer un modèle de requête
- L'enregistrer sur le SGBD le temps du script
- Instanciation du modèle et exécution de la requête

- **Avantages:**

- Cas de requêtes multiples
- Sécurité: éviter les attaques de type "injection SQL" (le SGBD sait ce qu'il doit recevoir; il vérifie les données transmises et fait les échappements nécessaires)

- **Inconvénient:**

- Temps légèrement plus long, si 1 seule fois la requête

Requêtes "préparées"

- **Construction du modèle de requête :**

```
$sql="INSERT INTO ville (nomv,pays,descripville)  
values (:nomv, :pays, :descripville)";
```

- **Préparation du modèle de requête par le SGBD :**

```
$stmt=$dbh->prepare($sql);
```

- **Exécution de la requête :**

```
$nomv="O'Sullivan";
```

```
$pays='Irlande';
```

```
$descripville="belle ville!";
```

```
$stmt->BindParam(':nomv',$nomv);
```

```
$stmt->BindParam(':pays',$pays);
```

```
$stmt->BindParam(':descripville',$descripville);
```

```
$stmt->execute();
```

Requêtes "préparées"

```
$nomv="O'Sullivan";  
$req="SELECT nomv,pays FROM ville where nomv=:nomv";  
$stmt=$dbh->prepare($req);  
$stmt->execute(array(':nomv'=>$nomv));  
while ($row = $stmt->fetch()) {  
    echo $row['nomv'];  
    echo " - ";  
    echo $row['pays'];  
    echo '<br />';  
}  
$req="DELETE FROM ville where nomv=:nomv";  
$stmt=$dbh->prepare($req);  
$stmt->execute(array(':nomv'=>$nomv));
```

Mise en pratique

- tp2 sur l'intranet
- Changer le mode de gestion des données dans l'application image en passant à une base de données. Seule la partie Modèle est modifiée.
- Utiliser PDO pour communiquer avec le SGBD