PHP et accès à une base de données

Table des matières

1. Contexte

2. Architecture PHP/BD

Exemple

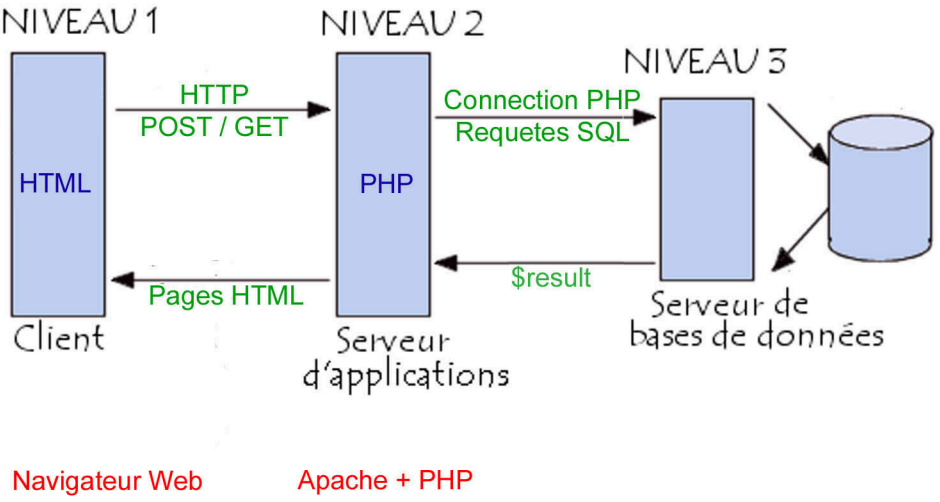


Image 1 Exemple d'architecture 3-tiers : PHP/BD

3. Exercice

TODO

4. PHP Data Objects

Az Définition

« PDO fournit une interface d'abstraction à l'accès de données, ce qui signifie que vous utilisez les mêmes fonctions pour exécuter des requêtes ou récupérer les données quelque soit la base de données utilisée. »

http://www.php.net/manual/fr/intro.pdo.php

Connexion à PostgreSQL avec PDO en PHP

Syntaxe

```php
$conn = new PDO('pgsql:host=hostname;port=5432;dbname=db', 'user', 'pass');
```

Exécution de requête SQL

Syntaxe

```php
$sql = '...';
$resultset = $connexion->prepare($sql);
$resultset->execute();
```

Traitement de résultat de requête SQL

Syntaxe

```php
while ($row = $resultset->fetch(PDO::FETCH_ASSOC)) {
  ... $row['...'];
  }

```

Complément

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

5. Exercice

TODO

6. Requêtes préparées et paramétrées

Requête préparée et paramétrée — Az Définition

L'étape de préparation de requête permet de précompiler une requête SQL côté serveur. Cela permet d'optimiser les performances lorsqu'une requête est utilisée plusieurs fois.

Cela permet également de paramétrer les requêtes au niveau d'un langage applicatif (comme PHP) sans utiliser de concaténation de chaîne (ce qui renforce notamment la sécurité du code).

Conseil

Utiliser des requêtes paramétrées au niveau des langages applicatifs.

Requête INSERT non préparée en PHP (déprécié) — Exemple

```php
$result = $connexion->query("INSERT INTO medicament (nom) VALUES ('Nouveau')");
```

Requête INSERT préparée en PHP — Exemple

```php
$result = $connexion->prepare("INSERT INTO medicament (nom) VALUES
('Nouveau')");
$result->execute();
```

Requête INSERT préparée et paramétrée en PHP (bindValue) — Exemple

```php
$result = $connexion->prepare("INSERT INTO medicament (nom) VALUES (:param1)");
$result->bindValue('param1', $name, PDO::PARAM_STR);
$result->execute();

```

Requête INSERT préparée et paramétrée en PHP (bindParam) — Exemple

```php
$result = $connexion->prepare("INSERT INTO medicament (nom) VALUES (:param1)");
$result->bindParam(':param1', $name, PDO::PARAM_STR);

$name = "Nouveau1";
$result->execute();

$name = "Nouveau2";
$result->execute();
```

Requête préparée en PHP

Complément

http://php.net/manual/fr/pdo.prepared-statements.php

7. À l'école de musique

[30 min]

Une école de musique souhaite gérer les inscriptions aux différentes classes d'instrument et aux orchestres à l'aide d'une base de données.

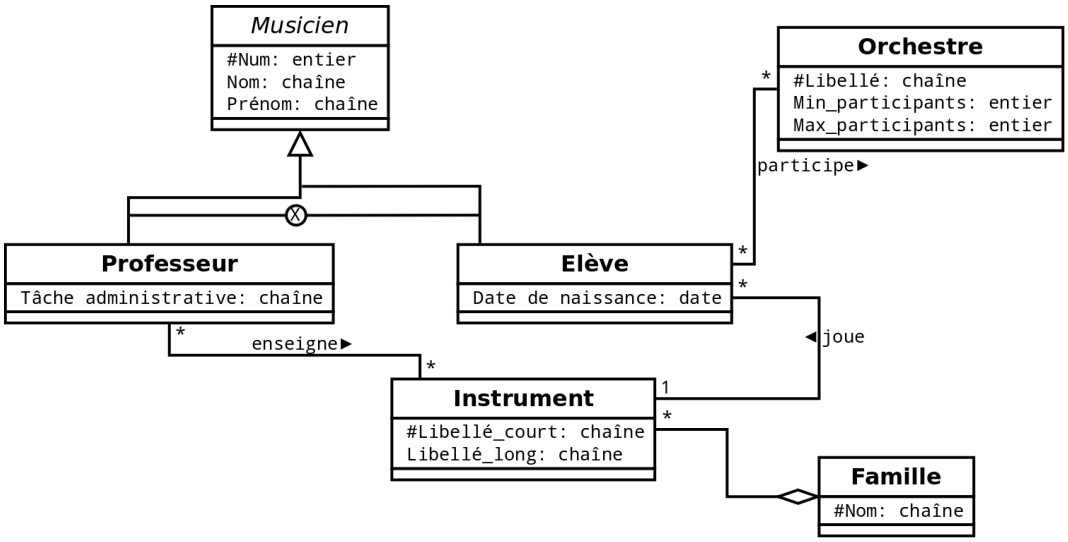


Schéma conceptuel

```
Famille (#nom:chaîne)
Instrument (#lib:chaîne, lib_long:chaîne, famille=>Famille(nom))
Professeur (#num:entier, nom:chaîne, prénom:chaîne, tâche:chaîne)
Eleve (#num:entier, nom:chaîne, prénom:chaîne, date_naissance:date,
inst=>Instrument(lib))
Orchestre (#lib:chaîne, min:entier, max:entier)
Enseigne (#num=>Professeur(num), #lib=>Instrument(lib))
Participe (#num=>Eleve(num), #lib=>Orchestre(lib))
```

Question 1

[solution n°1 p. 19]

Créer la base de données permettant de gérer les **élèves** et les **instruments** (sans les familles). Insérer des données exemple.

Question 2

[solution n°2 p. 19]

Écrire le code SQL qui permet d'afficher la liste des élèves (nom et prénom) triés par instrument.

Question 3

[solution n°3 p. 19]

Écrire le code SQL qui permet d'afficher la liste des instruments avec le nombre d'élèves associés (on affichera même les instruments dont personne ne joue).

Question 4

[solution n°4 p. 19]

Écrire le code SQL qui permet d'afficher la liste des instruments avec le nombre de places restantes par instruments, sachant qu'il y a 20 places disponibles par instrument.

Question 5

[solution n°5 p. 19]

Compléter le code PHP ci-dessous pour afficher le nombre de places disponibles par instruments listés par leur libellé long.

On utilisera une requête paramétrée.

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>École de musique</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  </head>

  <body>
  <h2>Liste des places disponibles par instrument</h2>
  <table border="1">
  <tr> <td><b>Instruments</b></td> <td><b>Places disponibles</b></td> </tr>

  <?php
  /* Nombre maximum d'élèves autorisé par instrument */
    $max_eleves = 20;

    /* Nombre maximum d'élèves autorisé par instrument */
    $connexion = new PDO('pgsql:host=localhost;port=5432;dbname=musique', 'me',
'mypassword');

    /** Préparation et exécution de la requête **/
    $sql = 'SELECT I.lib AS lib, ? - COUNT(E.inst) AS dispo ...';
    $resultset = $connexion->prepare($sql);
    $resultset->bindParam(1, ...);
    $resultset->execute();

    while ($row = $resultset->fetch(PDO::FETCH_ASSOC)) {
        echo '<tr>';
        echo '<td>' . $row[...] . '</td>';
        echo '<td>' . $row[...] . '</td>';
        echo '</tr>';
        }

    /** Déconnexion **/
    $connexion=null;
    ?>

    </table>
    </body>
</html>
```

Indice :

Requêtes préparées (cf. p.8)

8. Accès à une BD en écriture (INSERT, UPDATE, DELETE)

Exemple

```php
<?php

/** Connexion **/
$connexion = new PDO('pgsql:host=localhost;port=5432;dbname=test', 'test', 'test');

/** Préparation et exécution de la requête **/
$sql = 'INSERT INTO medicament (nom) VALUES (\'Nouveau\')';
$result = $connexion->prepare($sql);
$result->execute();

/** Traitement du résultat **/
if ($result) {
  echo 'Nouveau inséré';
}
else {
  echo 'Erreur lors de l\'insertion';
}

/** Déconnexion **/
$connexion=null;

?>
```

9. Super-transferts

L'entreprise de ventes de figurines de super-héros GARVEL a monté un partenariat avec les deux sites de ventes en ligne *makemoney.com* et *dobusiness.com*. Chaque entreprise lui demande de mettre à disposition respectivement un fichier CSV et un fichier XML pour le transfert du catalogue, stocké dans une base de données PostgreSQL.

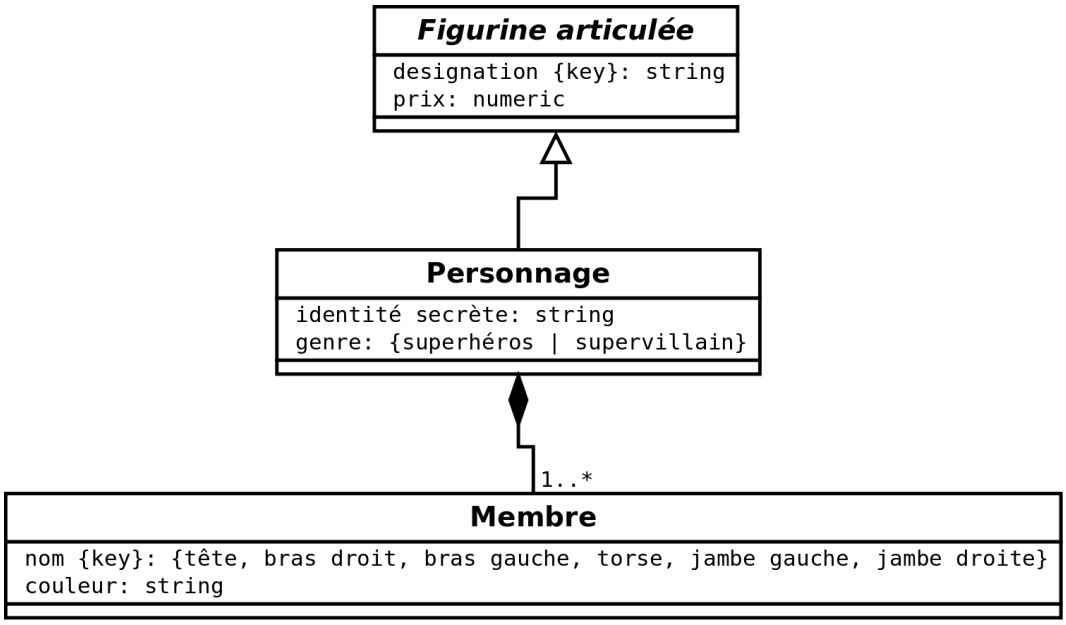
Le code devra être exécuté et testé.

Fichier CSV et fichier XML requis

```
Superman;15
Batman;12
Superchild;12
...
```

```
<catalogue>
  <figurine designation='Superman' prix='1555'/>
  <figurine designation='Batman' prix='12'/>
  <figurine designation='Superchild' prix='12'/>
  ...
</catalogue>
```

Modèle de la base de données



Modèle UML Figurines GARVEL (extrait)

Question 1

[solution n°6 p. 20]

Créer la base de données correspondant à ce modèle.

Créer une vue `vfigurine` permettant de retourner les champs `designation` et `prix`.

Créer un utilisateur `customer` permettant de lire la vue figurine.

Question 2

[solution n°7 p. 20]

Réaliser un script PHP *csv.php* permettant de se connecter à la base PosgreSQL et d'afficher la désignation et le prix au format CSV, en suivant l'exemple ci-après.

Indice :

Accès à une BD en lecture (SELECT) *(cf. p.15)*

Question 3

[solution n°8 p. 21]

Réaliser un script PHP permettant de se connecter à la base et d'afficher la désignation et le prix selon un schéma XML, en suivant l'exemple ci-après.

10. Accès à une BD en lecture (SELECT)

Exemple

```php
<?php

/** Connexion **/
$connexion = new PDO('pgsql:host=localhost;port=5432;dbname=test', 'test',
'test');

/** Préparation et exécution de la requête **/
$sql = 'SELECT nom FROM medicament';
$resultset = $connexion->prepare($sql);
$resultset->execute();

/** Traitement du résultat **/
while ($row = $resultset->fetch(PDO::FETCH_ASSOC)) {
  echo $row['nom'];
  echo ' ';
}

/** Déconnexion **/
$connexion=null;

?>
```

11. Auto-évaluation

11.1. Exercice final

Quiz - Culture

[solution n°9 p. 21]

Exercice

...

A A

B B

Quiz - Méthode

[solution n°10 p. 22]

Exercice

...

A A

B B

Quiz - Code

[solution n°11 p. 22]

Exercice

...

A A

B B

11.2. Devoirs en ligne

[45 min]

Une université propose des formations pour ses étudiants via une plate-forme d'enseignement en ligne. Plusieurs devoirs sont proposés, chacun ayant une description et une même date de rendu pour tous les étudiants souhaitant le faire. Le but de cet exercice est de réaliser un site Web permettant aux étudiants de consulter leurs notes.

Base de données

On vous donne ci-dessous le code SQL LDD de la base de données sous PostgreSQL ainsi qu'un exemple de code SQL LMD pour l'insertion de données.

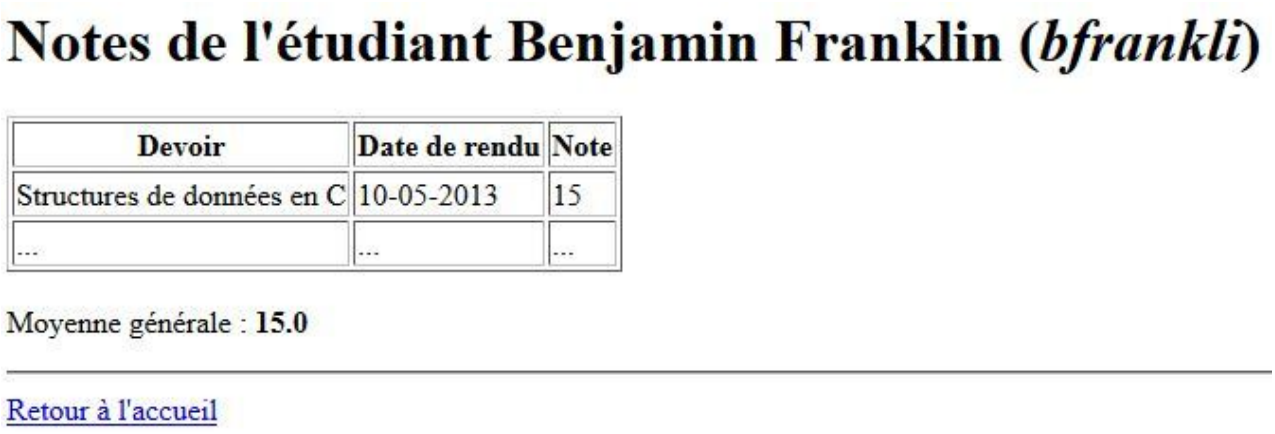
```sql
CREATE TABLE etudiant(
login CHAR(8) PRIMARY KEY,
nom TEXT,
prenom TEXT
);

CREATE TABLE devoir(
num INTEGER PRIMARY KEY,
daterendu DATE UNIQUE NOT NULL,
description TEXT NOT NULL
);

CREATE TABLE note(
etudiant CHAR(8) REFERENCES etudiant(login),
devoir INTEGER REFERENCES devoir(num),
valeur INTEGER NOT NULL,
PRIMARY KEY(etudiant, devoir),
CHECK (valeur BETWEEN 0 AND 20)
);

INSERT INTO etudiant(login, nom, prenom) VALUES ('bfrankli', 'Franklin', 'Benjamin');
INSERT INTO devoir(num, daterendu, description) VALUES (1, '10-05-2013','Structures de donnees en C');
INSERT INTO note(etudiant, devoir, valeur) VALUES ('bfrankli', 1, 15);
```

Site Web

Le site Web sera composé :

- d'une page d'accueil (`accueil.html`) comportant un formulaire où l'étudiant doit entrer son login ;
- d'une page présentant les notes d'un étudiant pour chaque devoir qu'il a rendu ainsi que sa moyenne générale (`notes.php`).

La seconde page (`notes.php`) est appelée à partir de la première (`accueil.html`). On supposera dans le reste de l'exercice que tous les fichiers sont situés directement dans le répertoire `public_html` du serveur (sans sous-répertoires).



Exemple d'affichage pour notes.php

Question 1

[solution n°12 p. 22]

Écrivez le formulaire HTML de la page d'accueil (`accueil.html`). On utilisera la méthode HTTP POST.

Indice :

Requête GET ou POST par formulaire HTML (balise <form>)

```
1 <form method="post" action="notes.php">
2   <p>Login : </p>
3   <input type="text" name="login"/>
4   <input type="submit"/>
5 </form>
```

Dans les questions suivantes, on supposera que le login spécifié dans le formulaire de `accueil.html` existe effectivement dans la base de données (on ne fera donc pas de test pour le vérifier) et on se référera à l'exemple d'affichage pour `notes.php` pour le résultat souhaité.

Question 2

[solution n°13 p. 22]

Écrivez le code PHP permettant de générer le code HTML du titre de la page "Notes de l'étudiant..." pour le login en question.

Indice :

Traiter les requêtes HTTP avec un serveur PHP

Indice :

```
1 echo "<h1>Notes de l'étudiant $vRow[prenom] $vRow[nom] (<i>$vRow[login]</i>)</h1>";
```

Question 3

[solution n°14 p. 23]

Écrivez le code PHP permettant de générer le code HTML du tableau des notes de l'étudiant. Les devoirs seront affichés par ordre croissant de date de rendu.

Indice :

On ajoutera une vue à la base de données préalablement à l'écriture du code PHP, afin d'éviter que la couche PHP ne contiennent du code SQL avancé.

Question 4

[solution n°15 p. 23]

Écrivez le code PHP permettant de calculer la moyenne générale de l'étudiant.

Question 5

[solution n°16 p. 23]

Écrivez le code HTML du lien hypertexte "Retour à l'accueil" et finalisez la page `notes.php`.

Indice :

```
1 <a href="accueil.html">Retour à l'accueil</a>
```

Solutions des exercices

Solution n°1

[exercice p. 10]

```sql
CREATE TABLE instrument (
lib VARCHAR PRIMARY KEY,
lib_long VARCHAR
);

CREATE TABLE eleve (
num INTEGER PRIMARY KEY,
nom VARCHAR NOT NULL,
prenom VARCHAR,
date_naissance DATE,
inst VARCHAR REFERENCES instrument(lib)
);

INSERT INTO instrument (lib) VALUES ('Violon');
INSERT INTO instrument (lib) VALUES ('Guitare');

INSERT INTO eleve (num, nom, inst) VALUES (1, 'Jimmy', 'Guitare');
INSERT INTO eleve (num, nom, inst) VALUES (2, 'Robby', 'Guitare');
```

Solution n°2

[exercice p. 10]

```sql
SELECT E.nom, E.prenom, E.inst
FROM Eleve E
ORDER BY E.inst ;
```

Solution n°3

[exercice p. 10]

```sql
SELECT I.lib, COUNT(E.inst)
FROM Instrument I LEFT JOIN Eleve E ON E.inst=I.lib
GROUP BY lib ;
```

Solution n°4

[exercice p. 10]

```sql
SELECT I.lib, 20 - COUNT(E.inst)
FROM Instrument I LEFT JOIN Eleve E ON E.inst=I.lib
GROUP BY lib ;
```

Solution n°5

[exercice p. 11]

```html
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>École de musique</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  </head>

  <body>
  <h2>Liste des places disponibles par instrument</h2>
  <table border="1">
```

```
  <tr> <td><b>Instruments</b></td> <td><b>Places disponibles</b></td> </tr>

  <?php
  /* Nombre maximum d'élèves autorisé par instrument */
    $max_eleves = 20;

    /* Nombre maximum d'élèves autorisé par instrument */
    $connexion = new PDO('pgsql:host=localhost;port=5432;dbname=test', 'test',
'test');

    /** Préparation et exécution de la requête **/
    $sql = 'SELECT I.lib AS lib, :max - COUNT(E.inst) AS dispo
            FROM Instrument I LEFT JOIN Eleve E ON E.inst=I.lib
            GROUP BY lib';
    $resultset = $connexion->prepare($sql);
    $resultset->bindParam('max', $max_eleves);
    $resultset->execute();

    while ($row = $resultset->fetch(PDO::FETCH_ASSOC)) {
        echo '<tr>';
        echo '<td>' . $row['lib'] . '</td>';
        echo '<td>' . $row['dispo'] . '</td>';
        echo '</tr>';
        }

    /** Déconnexion **/
    $connexion=null;
    ?>

    </table>
    </body>
</html>
```

Solution n°6

[exercice p. 13]

```
CREATE TABLE Personnage (
designation VARCHAR PRIMARY KEY,
prix DECIMAL NOT NULL,
identite VARCHAR,
genre VARCHAR(12) CHECK (genre='superhéros' OR genre='supervillain')
);

INSERT INTO Personnage (designation, prix) VALUES ('Superman',15);
INSERT INTO Personnage (designation, prix) VALUES ('Batman',12);
INSERT INTO Personnage (designation, prix) VALUES ('Superchild',12);

CREATE VIEW vfigurine AS
SELECT designation, prix FROM Personnage;

CREATE USER customer WITH ENCRYPTED PASSWORD 'public';
GRANT SELECT ON vfigurine TO customer;
```

Solution n°7

[exercice p. 14]

```
<?php

/** Connexion **/
$connexion = new PDO('pgsql:host=localhost;port=5432;dbname=test', 'test', 'test');

/** Préparation et exécution de la requête **/
```

```php
$sql = 'SELECT designation, prix FROM vfigurine';
$resultset = $connexion->prepare($sql);
$resultset->execute();

/** Traitement du résultat **/
while ($row = $resultset->fetch(PDO::FETCH_ASSOC)) {
  echo $row['designation'] . ";" . $row['prix'] . "\n";
}

/** Déconnexion **/
$connexion=null;

?>
```

Solution n°8

[exercice p. 14]

```php
<?php

/** Connexion **/
$connexion = new PDO('pgsql:host=localhost;port=5432;dbname=garvel', 'customer', 'public');

/** Préparation et exécution de la requête **/
$sql = "SELECT designation, prix FROM vfigurine";
$resultset = $connexion->prepare($sql);
$resultset->execute();

/** Traitement du résultat **/
echo "<catalogue>";
while ($row = $resultset->fetch(PDO::FETCH_ASSOC)) {
  echo "<figurine designation='" . $row['designation'] . "' prix='" . $row['prix'] . "'/>";
}
echo "</catalogue>";

/** Déconnexion **/
$connexion=null;

?>
```

Solution n°9

[exercice p. 16]

Exercice

...

- [x] A A
- [ ] B B

Solution n°10

[exercice p. 16]

Exercice

...

A A ✓

B B

Solution n°11

[exercice p. 16]

Exercice

...

A A ✓

B B

Solution n°12

[exercice p. 18]

```
<html>
<body>
<h1>Accueil</h1>
<form method="post" action="notes.php">
  <p>Login : </p>
  <input type="text" name="login"/>
  <input type="submit"/>
</form>
</body>
</html>
```

Solution n°13

[exercice p. 18]

```
<?php
// Connexion à la base de données
$vConn = new PDO('pgsql:host=localhost;port=5432;dbname=devoirs', 'test', 'test');

// Écriture, préparation et exécution de la requête 1
$vSql = 'SELECT login, nom, prenom FROM etudiant WHERE login=:login';
$vResultSet = $vConn->prepare($vSql);
$vResultSet->bindValue(':login',$_POST['login'],PDO::PARAM_STR);
$vResultSet->execute();
// Traitement du résultat
$vRow = $vResultSet->fetch(PDO::FETCH_ASSOC);
echo "<h1>Notes de l'étudiant $vRow[prenom] $vRow[nom] (<i>$vRow[login]</i>)</h1>";

// Clôture de la connexion
$vConn=null;
?>
```

Solution n°14

[exercice p. 18]

SQL

```sql
CREATE VIEW v_devoir AS
SELECT d.description, d.daterendu, n.etudiant, n.valeur
FROM devoir d JOIN note n ON n.devoir=d.num
ORDER BY d.daterendu;
```

PHP

```php
// Écriture, préparation et exécution de la requête 2
$vSql = 'SELECT * FROM v_devoir WHERE etudiant=:login';
$vResultSet = $vConn->prepare($vSql);
$vResultSet->bindValue(':login',$_POST['login'],PDO::PARAM_STR);
$vResultSet->execute();
// Traitement du résultat
echo "<table border='1'>";
echo "<tr><th>Devoir</th><th>Date de rendu</th><th>Note</th></tr>";
while ($vRow = $vResultSet->fetch(PDO::FETCH_ASSOC)) {
  echo "<tr><td>$vRow[desc]</td><td>$vRow[date]</td><td>$vRow[note]</td></tr>";
}
echo "</table>";
```

Solution n°15

[exercice p. 18]

SQL

```sql
CREATE VIEW v_moy AS
SELECT ROUND(AVG(valeur),1) AS moy, etudiant
FROM note
GROUP BY etudiant;
```

```php
// Écriture, préparation et exécution de la requête 3
$vSql = 'SELECT * FROM v_moy WHERE etudiant=:login';
$vResultSet = $vConn->prepare($vSql);
$vResultSet->bindValue(':login',$_POST['login'],PDO::PARAM_STR);
$vResultSet->execute();
// Traitement du résultat
$vRow = $vResultSet->fetch(PDO::FETCH_ASSOC);
echo "<p>Moyenne générale : <b>$vRow[moy]</b></p>";
```

Solution n°16

[exercice p. 18]

```php
<?php
// Connexion à la base de données
$vConn = new PDO('pgsql:host=localhost;port=5432;dbname=test', 'test', 'test');

// Écriture, préparation et exécution de la requête 1
$vSql = 'SELECT login, nom, prenom FROM etudiant WHERE login=:login';
$vResultSet = $vConn->prepare($vSql);
$vResultSet->bindValue(':login',$_POST['login'],PDO::PARAM_STR);
$vResultSet->execute();
// Traitement du résultat
$vRow = $vResultSet->fetch(PDO::FETCH_ASSOC);
echo "<h1>Notes de l'étudiant $vRow[prenom] $vRow[nom] (<i>$vRow[login]</i>)</h1>";

// Écriture, préparation et exécution de la requête 2
$vSql = 'SELECT * FROM v_devoir WHERE etudiant=:login';
$vResultSet = $vConn->prepare($vSql);
$vResultSet->bindValue(':login',$_POST['login'],PDO::PARAM_STR);
$vResultSet->execute();
```

```php
// Traitement du résultat
echo "<table border='1'>";
echo "<tr><th>Devoir</th><th>Date de rendu</th><th>Note</th></tr>";
while ($vRow = $vResultSet->fetch(PDO::FETCH_ASSOC)) {
  echo "<tr><td>$vRow[desc]</td><td>$vRow[date]</td><td>$vRow[note]</td></tr>";
}
echo "</table>";

// Écriture, préparation et exécution de la requête 3
$vSql = 'SELECT * FROM v_moy WHERE etudiant=:login';
$vResultSet = $vConn->prepare($vSql);
$vResultSet->bindValue(':login',$_POST['login'],PDO::PARAM_STR);
$vResultSet->execute();
// Traitement du résultat
$vRow = $vResultSet->fetch(PDO::FETCH_ASSOC);
echo "<p>Moyenne générale : <b>$vRow[moy]</b></p>";

// Lien de retour
echo "<p><a href='accueil.html'>Retour à l'accueil</a></p>";

// Clôture de la connexion
$vConn=null;
?>
```

Index

Crédits des ressources

Exemple d'architecture 3-tiers : PHP/BD p. 4
commentcamarche.net - 2003 Pillou - GNU FDL