Sujet 7 - Technologies du Web

mars 2020

Web et base de données

Cette fiche est consacrée à l'utilisation d'une base de données depuis un projet web. Voici quelques uns des principes de l'organisation de la conception du code qu'il convient de respecter :

1. Le script (pour nous, en PHP) doit établir une connexion au SGBD. Cette connexion est persistante durant toute la durée d'exécution du script. Il faut établir **une seule connexion** et conserver un accès à cette connexion pendant toute l'exécution du script. En PHP la connexion est portée par un objet instance de la classe PDO. Il faudra instancier cet objet puis **le réutiliser** ensuite pour lancer les différentes requêtes.
2. Tout le code dédié à la l'interrogation et la manipulation de la base de données sera réuni en **un seul composant logiciel** qui constituera la **couche d'abstraction des données**. Les raisons à cela :
   - ne pas disséminer les requêtes SQL au sein du code PHP, pour la lisibilité et la bonne compréhension du code, ais aussi pour la maintenance.
   - rendre l'essentiel du code indépendant de la façon dont ou acquière et manipule les données
   - faciliter la modification de la gestion des données. Les éventuels changements de SGBD ou les évolutions de la structure des tables n'impacteront que la couche d'abstraction des données, pas le reste du code.
3. rappelons qu'une attention toute particulière doit être portée à la gestion du mot de passe (voir fiche précédente).

La couche d'abstraction de données sera implémentée par une **classe** nommée `DataLayer` destinée à être instanciée une seule fois (singleton). La connexion sera un attribut de la classe. Dans cette classe, on développera une méthode spécifique pour chaque besoin d'accès aux données.
Une première version de cette classe vous est donnée. Au fur et à mesure de l'exercice, vous aurez à la compléter.

Exercice 1 :
Prérequis : Il faut que la base de données « course cycliste » de la séance précédente soit opérationnelle sur le serveur Postgres.
Décompressez l'archive fournie et installez ces fichiers sur le serveur webtp.
Q 0 . Mise en place

1. comme indiqué dans l'exercice 2 de la fiche précédente, créez le fichier `lib/db_parms.php`.
2. ouvrez dans un éditeur `lib/DataLayer.class.php` pour en examiner le code. :
   - lors de l'instanciation, cette classe établit la connexion à la base de données ; l'objet représentant la connexion (instance de PDO) est conservé dans un attribut privé (`$connexion`). Durant l'exéction du script une seule instance de la classe DataLayer sera créée.
   - le rôle de la méthode `getCoureursQ0()` est d'interroger la B.D. (donc d'exécuter une requête SQL), de récupérer la réponse et de fournir le résultat sous forme d'une table PHP.
3. `testData.php` est un script de test qui instancie la couche de donnée, invoque `getCoureursQ1a()` et affiche le résultat brut sous forme de dump. Exécutez ce script et vérifiez que vous obtenez le résultat attendu (il figure en commentaire dans le script).

Q 1 . Améliorer et afficher la liste des coureurs
Dans cette question, nous allons améliorer la liste des coureurs et l'afficher dans une page web, en respectant l'organisation du code :

- dans le dossier `lib` : les définitions de fonctions et classes nécessaires
- dans la classe `DataLayer` (dossier `lib`) : la méthode qui interroge la base de données
- dans le dossier `views` : la composition de la page web

— à la racine du projet : le script invoqué directement par le client web. Ce script contrôle le fonctionnement de l'ensemble.

1. (a) ajoutez à la classe `DataLayer` une méthode `getCoureursQ1()` (voir sa spécification dans le fichier)
   (b) modifiez le script de test `testData.php` pour tester la nouvelle méthode. Vérifiez que vous obtenez le résultat attendu.
2. dans le fichier `lib/fonctionsHTML.php` créez la fonction `coureursToHTML($coureurs)` qui produit une table HTML représentant la liste des coureurs, à partir de la table PHP
3. ouvrez le fichier fourni `views/pageCoureurs.php` et examinez son contenu, en particulier ses spécifications (commentaires en tête)
4. faites de même pour le script fourni `question1.php` et testez son bon fonctionnement dans le navigateur.

Q 2 . La liste des équipes dans un formulaire.
Le but de cette question est de présenter un formulaire proposant la liste des équipes. Scripts intervenant dans cette question :

— `question2.php`, script principal (fourni)
— `views/pageEquipes.php` (fourni) : page affichée
— à compléter : `lib/DataLayer.class.php` et `lib/fonctionsHTML.php`

1. (a) ajoutez à la classe `DataLayer` une méthode `getEquipes()` (voir sa spécification dans le fichier)
   (b) modifiez le script de test `testData.php` pour tester la nouvelle méthode. Vérifiez que vous obtenez le résultat attendu.
2. dans le fichier `lib/fonctionsHTML.php` créez la fonction `equipeToOptionsHTML($equipe)` qui produit le code HTML d'une liste d'options. Chaque option propose une équipe. (voir spécification précise dans le fichier fourni)
3. ouvrez le fichier fourni `views/pageEquipes.php` et examinez son contenu, en particulier ses spécifications (commentaires en tête)
4. faites de même pour le script fourni `question2.php` et testez son bon fonctionnement dans le navigateur.

Q 3 . Afficher les informations détaillées sur une équipe.
But de la question : afficher les informations détaillées à propos d'une équipe (dont la liste de ses coureurs) et proposer le formulaire pour une nouvelle recherche.
Scripts intervenant dans cette question :

— `question3.php`, script principal (à créer en copiant `question2.php`) : appelé sans paramètre, il affichera simplement le formulaire ; appelé avec le paramètre `equipe`, il affichera également ses informations détaillées ou signalera une erreur dans le nom d'équipe, le cas échéant.
— `views/pageEquipes.php` (fourni) le script précédent qui utilisera maintenant aussi `views/components/sectionReponseEquipe.php`
— à compléter : `lib/DataLayer.class.php` et `lib/fonctionsHTML.php`

1. (a) ajoutez à la classe `DataLayer` une méthode `getInfoEquipe($equipe)` qui renvoie les informations complètes concernant une équipe ou FALSE si cette équipe n'existe pas. (voir sa spécification dans le fichier)

   Important : utilisez absolument une requête avec un « paramètre nommé » pour la valeur de l'équipe cherchée. Voir le transparent du cours n°36. Ne jamais injecter directement une variable dans une requête SQL :

   `"... where equipe=$equipe"` → `"... where equipe=:equipe"` + utiliser `bindValue()`

   (b) modifiez le script de test `testData.php` pour tester la nouvelle méthode. Vérifiez que vous obtenez le résultat attendu.
2. ajoutez à `views/lib/fonctionsHTML.php` une fonction `equipeToHTML($infoEquipe)`
3. dans `question3.php` , ajoutez le code nécessaire pour

— prendre en compte le paramètre HTTP (GET) `equipe` : une chaîne optionnelle éventuellement vide et copiez sa valeur dans une variable `$equipeChoisie`

— si le paramètre est présent et non vide, utilisez `getInfoEquipe()` pour vérifier que l'équipe existe et si oui obtenir les informations détaillées et les ranger dans une variable `$infoEquipe`

Q 4 . Aux informations affichées à la question précédente, on veut ajouter la liste des coureurs de l'équipe. Implémentez cette nouvelle version **en suivant la même méthodologie**. Vous créerez notamment dans DataLayer une méthode `getMembers($equipe)` qui renvoie la liste des coureurs de l'équipe.

Exercice 2 :

La course se déroule en plusieurs étapes. Nous allons compléter la base de données, pour représenter les différentes étapes et mémoriser les temps des coureurs. Tout d'abord vous allez ouvrir l'interface phpPgAdmin pour mettre à jour la base.

1. supprimez la table `etape` créée la dernière fois.
2. dans la fenêtre SQL exécutez les commandes suivantes (faites un copié/collé) :

```
create table etapes
  (numero serial, nom varchar(30),  primary key(numero));
create table releves
  (dossard smallint, etape smallint, chrono interval, primary key(dossard,etape));
alter table releves
  add check (chrono <='6h' and chrono>'0');
alter table releves
  add foreign key (etape) references etapes(numero);
alter table releves
  add foreign key (dossard) references coureurs(dossard);
```

La table `etapes` décrit les étapes de la course. Les étapes portent un nom et sont numérotées à partir de 1. Remarquez que le type de l'attribut `numero` est `SERIAL`. Ce type particulier définit un attribut `integer` qui sera **auto-incrémenté** : quand on crée une étape il est inutile (et même déconseillé) de préciser son numéro : le SGBD lui attribuera automatiquement le plus petit numéro non encore attribué. Une commande d'insertion typique ressemblera donc à
`insert into etapes(nom) values ('Contre la Montre')`
(NB : le type SERIAL n'est pas standard SQL mais tous les SGBD implémentent un mécanisme équivalent, éventuellement avec une autre syntaxe).

Q 1 . Réalisez une page destinée aux organisateurs de la course. Cette page affichera une liste des étapes et proposera deux opérations :

— créer une nouvelle étape à ajouter à la suite des autres. L'utilisateur doit préciser le nom de l'étape (chaîne non vide).

— effectuer une remise à zéro (suppression de toutes les étapes)

Quand l'utilisateur réalise une opération, la même page est ensuite ré-affichée (avec une liste mise à jour, évidemment). Le script principal s'appellera `gestionEtapes.php`. Implémentez-le en utilisant la méthodologie de l'exercice 1.

Attention : l'effacement du contenu de la table `etapes` demande normalement 3 opérations :

— pour la cohérence des données, il faut d'abord effacer tout le contenu de la table `releves` qui dépend de la table `etapes`. Commande : `delete from releves`

— l'effacement de la table elle-même : Commande : `delete from etapes`

— la remise à zéro du compteur des numéros. Commande : `alter sequence etapes_numero_seq restart`

Mais sachez que ces 3 opérations peuvent être réalisées en une seule commande SQL :
`truncate table etapes restart identity cascade`
Cette dernière solution est préférable.

Q 2 . Réalisez une page dédiée à la saisie des temps des coureurs. Un formulaire proposera de choisir une étape (parmi celle existante) et un dossard (parmi ceux existant) et d'indiquer le chrono

correspondant dans un champ texte ou de préciser que le coureur a abandonné au cours de l'étape (case à cocher).
Le chrono sera saisi dans un champ texte, sous la forme H :M :S (le nombre de secondes peut avoir une partie décimale). Par exemple `3:54:20.45`
NB : dans la base de données, un coureur ayant abandonné lors une étape figurera dans la table `releves`, mais sans valeur pour l'attribut temps (c'est à dire avec la valeur NULL en réalité).
Q 3 . Réalisez une page `stats.php` qui dresse la liste des étapes avec pour chacune son numéro, les chronos mini, maxi, moyen, le nombre de coureurs ayant pris le départ, le nombre de coureurs ayant terminé.
Une fois la page réalisée, faites la modification nécessaire pour afficher également le nom de chaque étape.
Q 4 . *« bonus » ;-)*
On souhaite afficher le tableau d'arrivée pour une étape donnée. Le tableau d'arrivée est la liste des coureurs, classés du premier au dernier arrivé. Pour chaque coureur, on affichera son dossard, son nom, son chrono, son équipe et la couleur de son maillot. Les coureurs ayant abondonné ne figurent évidemment pas dans le tableau.
Un formulaire proposera de choisir une étape (dans une liste) et affichera le tableau d'arrivée, selon le même principe de fonctionnement que pour les questions 2 et 3.
Les informations devront être extraites de la base en une seule requête SQL.
Cette question ne présente pas de difficultés PHP ou HTML : le schéma est similaire à celui des questions précédentes. Vous ferez d'abord la conception et la mise au point de la requête SQL en utilisant l'interface phpPgAdmin. Une fois la requête ainsi conçue et testée, vous la transposerez pour l'intégrer dans la partie "web".