Travaux Dirigés et Pratiques d'Architecture des Appli. de la toile

Michel Meynard

12 janvier 2010

1 Introduction, paramètres locaux

En cas de problème, veuillez lire les lignes suivantes avant de demander de l'aide. Ce TP est destiné à être exécuté dans un environnement comportant :

- un navigateur (client HTTP) ;
- un serveur HTTP configuré avec un module interprétant le PHP ;
- un serveur MySQL ;
- les droits suffisants pour écrire des pages PHP et manipuler des bases de données MySQL ;
- un outil d'administration phpMyAdmin.

Les paramètres actuels de ces outils dans le domaine `info-ufr.univ-montp2.fr` sont :

- `~/public_html` : le répertoire de vos documents web accessibles depuis n'importe quel serveur http. Vous y créerez un sous-répertoire `ArchiToile` pour y stocker les pages et scripts du TP. Les scripts php peuvent être situés n'importe où sous `public_html`, doivent avoir le suffixe `.php` et être **lisibles** par tous (`chmod a+r toto.php`).
- `http://localhost/~pdupont/ArchiToile` : l'URL pour accéder à vos pages et scripts (localhost, `127.0.0.1`, a1.info-ufr.univ-montp2.fr ou a1 étant le nom de votre machine : commande Unix `hostname`) ;
- `~/public_html/ArchiToile/index.html` : page d'accueil par défaut accessible par : `http://machine/~pdupont/ArchiToile`. Si cette page n'existe pas, alors la page `index.php` est recherchée ;
- DROITS :
  - vos répertoires doivent être traversables et lisibles : `chmod 755 public_html public_html/ArchiToile` ;
  - vos fichiers html doivent être lisibles : `chmod 644 public_html/index.html` ;
  - vos scripts php doivent être lisibles : `chmod 644 public_html/ArchiToile/hello.php` ;
  - vos cgi doivent être lisibles et exécutables : `chmod 755 monrep/moncgi`.
- MySQL : la machine exécutant le serveur `mysqld` est `venus` sur le port 3306 ;
- Pour s'y connecter en mode console : `mysql -h venus -u pdupont pdupont -p`
- phpMyAdmin : l'outil `phpMyAdmin` est également situé sur `venus`. Son URL est : `http://10.6.200.XXX/phpMyAdmin4/` ;

Vous pourrez trouver à l'adresse suivante `http://www.lirmm.fr/~meynard/ArchiToile/`, des exemples correspondant aux exercices suivants.

2 Formulaires HTML et PHP

Exercice 1 (Factorielle) On s'intéresse à la fonction factorielle définie par $0! = 1$ et $(n+1)! = (n+1) * n!$.

1. écrire la fonction factorielle en PHP ;
2. écrire en un seul fichier, une page permettant de saisir un nombre entier puis d'afficher sa factorielle ;
3. que se passe-t-il si le paramètre passé est incorrect ? Comment y remédier ?

Exercice 2 (Champs cachés et commande) On souhaite écrire une page permettant de calculer le montant d'une commande composées d'articles en quantités variables. Les articles et leur tarifs seront insérés dans un tableau PHP. Par exemple :

```
$larticle=array('marteau'=>10, 'tenaille'=>5, 'vis'=>5.2, 'clou'=>5.8,
   'tournevis'=>7, 'ciseau'=>4, 'toile émeri'=>3);
```

Le formulaire devra donc comprendre un nombre de lignes de commandes variables qui augmentera au fur et à mesure que le client commandera de nouveaux articles. A chaque fois que le client validera une nouvelle ligne de commande (nom article, quantité), le formulaire devra être mis à jour pour refléter la bonne quantité et un total à payer correct.

1. Comment conserver l'information sur les articles et leur quantité commandée sans utiliser de session, ni de cookie, ni de fichier, ni de BD ?
2. écrire en un seul fichier, une page permettant de saisir une commande.
3. comment ajouter la suppression de lignes de commandes ?

Exercice 3 (Session et Mastermind) En utilisant la notion de session PHP, vous créerez un site permettant de jouer au Mastermind. Le jeu crèe un code aléatoire caché à 4 chiffres différents. Le joueur tente à chaque coup une combinaison de 4 chiffres différents et le jeu lui répond en indiquant le nombre de chiffres bien placés et le nombre de chiffres mal placés. Vous conserverez la liste des coups joués ainsi que leur résultat dans la session PHP. Vous modéliserez le jeu par une classe PHP5 qui mémorise le code aléatoire ainsi que la liste des essais successifs et de leurs résultats.

1. Décrire l'interface de la classe Mastermind.
2. Comment stocker une instance de Mastermind dans une SESSION PHP ?
3. Programmer la classe Mastermind dans le fichier Mastermind.php.
4. Ecrire le script du jeu.

Exercice 4 (MySQL et Liste d'étudiants) On utilise une base de données MySQL d'étudiants possédant 3 tables ayant la structure suivante :

```
-- Structure de la table ‘etudiant‘
CREATE TABLE ‘etudiant‘ (
  ‘nom‘ varchar(20) NOT NULL default ’’,
  ‘prenom‘ varchar(20) NOT NULL default ’’,
  ‘statut‘ char(2) NOT NULL default ’FI’,
  ‘groupe‘ tinyint(1) NOT NULL default ’0’,
  ‘email‘ varchar(20) NOT NULL default ’’,
  ‘opt‘ enum(’B’,’S’,’C’,’L’,’W’) default NULL,
  ‘numStageA‘ tinyint(2) unsigned default ’0’,
  PRIMARY KEY  (‘nom‘,‘prenom‘),
  UNIQUE KEY ‘email‘ (‘email‘)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 PACK_KEYS=1;

-- Structure de la table ‘options‘
CREATE TABLE ‘options‘ (
  ‘code‘ char(1) NOT NULL default ’’,
  ‘nom‘ varchar(30) NOT NULL default ’’,
  ‘resp‘ varchar(30) NOT NULL default ’’,
  ‘email‘ varchar(30) NOT NULL default ’’,
  PRIMARY KEY  (‘code‘)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 PACK_KEYS=1;

-- Structure de la table ‘stageA‘
CREATE TABLE ‘stageA‘ (
  ‘numStageA‘ tinyint(2) unsigned NOT NULL default ’0’,
  ‘sujet‘ varchar(255) NOT NULL default ’’,
  ‘entreprise‘ varchar(50) NOT NULL default ’’,
  ‘lieu‘ varchar(150) NOT NULL default ’’,
  ‘respEnt‘ varchar(150) NOT NULL default ’’,
  ‘respPeda‘ varchar(150) NOT NULL default ’’,
  PRIMARY KEY  (‘numStageA‘),
  KEY ‘numStageA‘ (‘numStageA‘)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 PACK_KEYS=1;
```

Chaque étudiant est inscrit dans une option et dans un stage. Le contenu de la BD est visible à l'adresse : `http://www.lirmm.fr/~meynard/ArchiToile/crebas.sql.html`.

1. Créer les tables dans votre BD MySQL grâce à l'onglet d'importation de PhpMyAdmin.
2. Afficher la liste des noms d'étudiants grâce à un script PHP.
3. Ecrire un script PHP `stageA.php` permettant de regrouper les étudiants participant à un même stage d'analyse. Chaque stage sera présenté (numéro, sujet, responsable, tuteur, et enfin les étudiants y participant.
4. On souhaite visualiser les étudiants selon différents critères et écrire un formulaire PHP `trombino.php` comportant :
   - une liste des options avec choix multiples ;
   - une liste de l'odre d'affichage avec choix unique parmi nom et prénom, statut, groupe, option ;
   - un bouton de validation

   Les choix d'affichage devront être mémorisés.

Exercice 5 (Cookies et Sessions) Afin de bien comprendre les mécanismes de session et de cookie, nous allons commencer une partie de Mastermind sur le site du Lirmm : `http://www.lirmm.fr/~meynard/ArchiToile/master.php`. Après avoir joué un coup "1234" et avoir soumis le formulaire, réaliser les manipulations suivantes :

1. afficher les cookies du lirmm. Repérer PHPSESSID et récupérer sa valeur.
2. supprimer ce cookie ; jouer une nouvelle proposition "5678" : que se passe-t-il ?

3. afficher les cookies du lirmm. Repérer PHPSESSID et comparer sa valeur avec la précédente. Que s'est-il passé ?
4. Refuser les cookies et tenter de jouer une partie. Que se passe-t-il ?
5. Passer le premier PHPSESSID comme paramètre de l'URI : `http://www.lirmm.fr/~meynard/ArchiToile/master.php?PHPSESSID=0c92dbddd4d1ada27ea223b0bc651ff2`. Que se passe-t-il ?
6. Si l'on fait maintenant une nouvelle proposition et que l'on clique sur OK, le premier jeu continue-t-il ?
7. Accepter à nouveau les cookies. Jouer plusieurs nouveaux coups, lancer une nouvelle partie et observer les cookies du lirmm : qu'en déduisez-vous ?
8. Saisissez l'URL `http://www.lirmm.fr/~meynard/ArchiToile/master.php` dans le navigateur. Que se passe-t-il le premier coup et le second coup joués ?
9. Fermer le navigateur et en relancer un autre.Saisissez l'URL : `http://www.lirmm.fr/~meynard/ArchiToile/master.php?PHPSESSID=0c92dbddd4d1ada27ea223b0bc651ff2` avec un PHPSESSID précédent. Retrouve-t-on l'ancienne partie ?

Exercice 6 (Cookies et Mastermind) Réécrire le jeu de Mastermind en ajoutant la fonctionalité suivante : un joueur peut sauvegarder une partie en cours dans un cookie en lui donnant un nom de son choix. Lorsqu'il joue, il peut sauver la partie en cours, restaurer une ancienne partie, jouer une nouvelle partie, ou jouer un coup. Les cookies devront avoir une durée de vie d'un jour. Faites des essais en affichant les cookies, en les supprimant, ...

Exercice 7 (Authentification avec MySQL) A l'aide de PhpMyAdmin, créer une table : `utilisateur(login varchar(30), pasword varchar(50), nom ...)`. Ajouter quelques utilisateurs à cette table en ayant soin de crypter avec la fonction `md5()` le mot de passe de chaque utilisateur. Créer un script PHP qui nécessite l'authentification d'un utilisateur au long des différentes pages du site. L'authentification sera réalisée par un formulaire qui une fois posté et vérifié, sauvegardera le login dans la variale de SESSION.

Sur la page `http://www.lirmm.fr/~meynard/ArchiToile/authentif.php`, on peut se connecter sous deux identités :

- etu, etu ;
- et prof, prof ;

Exercice 8 (Creole et Propel) Les questions suivantes sont à réaliser dans **votre environnement UFR** ! Les fichiers présentés correspondent à une **autre configuration** (Windows, Wamp, MySQL, en localhost). Ils vous faut donc effectuer les modifications dans chaque fichier de configuration.

1. tester l'installation de pear à la ligne de commande puis installer pear en hébergement mutualisé ;
2. installer les paquets creole, phing, propel `1.2` ...
3. Créer un répertoire pour votre projet Propel "MonPropel". Tous les fichiers suivants devront être dans ce répertoire.
4. Ecrire à la main le fichier `schema.xml` suivant :

```
<?xml version="1.0" encoding="utf-8"?>
<database name="cmd">
<table name="client">
  <column name="id" type="INTEGER" required="true" primaryKey="true" />
  <column name="nom" type="varchar" size="40" required="true" description="Nom du client"/>
  <column name="prenom" type="varchar" size="40" description="PrÃ©nom du client"/>
</table>

<table name="cmd">
  <column name="id" type="INTEGER" required="true" primaryKey="true" />
  <column name="date" type="timestamp" required="true" description="Date de la commande"/>
  <column name="client_id" type="INTEGER" required="true" description="id du client"/>
  <foreign-key foreignTable="client" onDelete="restrict" onUpdate="cascade">
      <reference local="client_id" foreign="id"/>
   </foreign-key>
</table>
</database>
```

5. Ecrire à la main le fichier `build.properties` suivant :

```
propel.project = cmd
propel.database = mysql
propel.database.url = mysql://root@localhost/cmd
```

6. Lancez la génération du code sql : `propel-gen MonPropel sql`. Puis examinez le fichier `MonPropel/build/sql/schema.sql`
7. Lancez la génération du modèle objet : `propel-gen MonPropel om`. Puis examinez les fichiers `MonPropel\build\classes\cmd\om`
8. Lancez la création des tables dans la bd grâce à la commande suivante : `propel-gen.bat MonPropel insert-sql`. Vérifiez grâce à PhpMyAdmin que les deux tables ont bien été créées.
9. Ecrire à la main le fichier `MonPropel/runtime.xml` suivant :

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<config>
 <log>
  <ident>cmd</ident>
  <level>7</level>
 </log>
 <propel>
  <datasources default="cmd">
   <datasource id="cmd">
    <!-- the Propel adapter will usually be the same as phptype of connection DSN -->
    <adapter>mysql</adapter>
    <connection>
     <phptype>mysql</phptype>
     <database>cmd</database>
     <hostspec>localhost</hostspec>
     <username>root</username>
<!--      <password></password> -->
    </connection>
   </datasource>
  </datasources>
 </propel>
</config>
```

. Ce dernier permet d'indiquer la configuration à l'exécution.

10. Lancez la création de la configuration à l'exécution en php grâce à la commande suivante : `propel-gen.bat MonPropel convert-props`. Vérifiez le fichier `MonPropel/build/conf/cmd-conf.php`. Remarquons que les cibles sql, om et convert-props peuvent être demandées plus simplement avec la commande simplifiée : `propel-gen.bat MonPropel`.
11. Pour utiliser le modèle objet, il suffit maintenant d'écrire le fichier suivant :

```
<?php
set_include_path("build/classes" . PATH_SEPARATOR . get_include_path());
include_once 'propel/Propel.php';
Propel::init("build/conf/cmd-conf.php");

include_once 'cmd/Client.php';
$dup=new Client();
$dup->setNom("Dupont");
$dup->save();
echo "client créé";
?>
```

Lancer ce fichier depuis un navigateur et vérifiez que le client a bien été créé dans la BD !

12. Utiliser PhpMyAdmin afin de réaliser les transformations suivantes sur la BD :
    - transformer le moteur de stockage en innoDB afin de permettre les transactions ;
    - créer la table article(id int autoinc, designation varchar(50), prix decimal(12,2)) ;
    - créer la table ligne(cmd_id int, article_id int , qte int) ;
    - ajouter les contraintes d'intégrité référentielle (gestion des relations) suivantes : cmd_id ref cmd(id), article_id ref article(id), client_id ref client(id)
    - peupler les tables avec des clients, des commandes, des articles, des lignes de commandes de quantité d'articles.
13. après avoir sauvé `schema.xml` dasn un autre fichier, recréez-le en utilisant la commande propel : `propel-gen.bat MonPropel creole`. Editez le nouveau schéma et remplacez les RESTRICT et CASCADE en restrict et cascade ; de même, supprimer les valeurs par défaut vide qui n'ont pas lieu d'exister (bugs version 1.2) ;
14. regénérez tout le modèle objet et vérifiez les nouvelles classes. S'il y a des erreurs, déboguez-les !