Inf340 Systèmes d'information

Premier site – ce qu'il ne faut pas faire

Objectifs

Objectif :

- Réaliser un premier site en utilisant une mauvaise approche de conception.
- Le site permet de gérer un carnet d'adresse composé d'une liste de noms et de numéros de téléphone, un nom peut posséder plusieurs numéros de téléphones.

Conditions du développement

Nous allons réaliser notre site web dynamique sur une machine de développement nous utiliserons WAMP :

- Windows
- Apache (serveur HTTP)
- MySQL (SGBD)
- PHP

Comme environnement de développement nous utiliserons eclipse

La mise en place du serveur apache

Ne souhaitant pas gérer de résolution de nom (serveur DNS, fichier « hosts »), nous allons utiliser un alias apache.

L'URL http://localhost/demo1 correspondra au répertoire D:\xxx\inf340_cm_demo1

L'assistant graphique de WAMP vous génère :
Alias /demo1/ "D:\xxx\inf340_cm_demo1/"

La mise en place du serveur apache

Vous devez obtenir une configuration de ce type

```
Alias /demo1/ "D:\xxx\inf340_cm_demo1/"

<Directory "D:\xxx\inf340_cm_demo1/">
    Options Indexes FollowSymLinks MultiViews
   AllowOverride all
   Require local

</Directory>
```

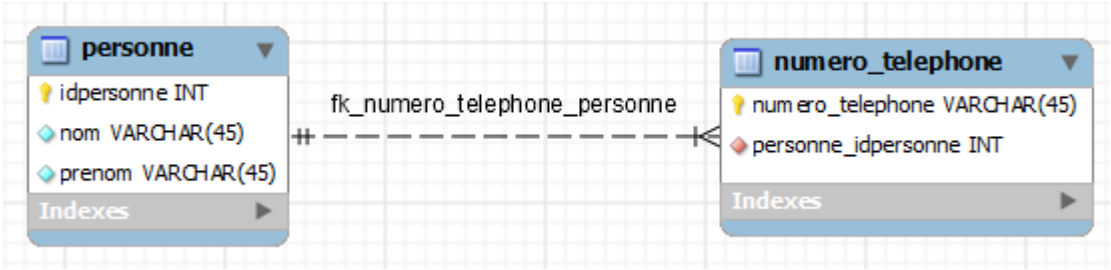
Définition du modèle

- Nous allons devoir rendre nos données persistantes et pour cela utiliser un SGBD.

- Pour nous il s'agit de MySQL.

- Pour créer les tables, vous pouvez utiliser :
  - des ateliers de génie logiciel comme PowerAMC qui à partir du MCD (Modèle Conceptuel de Données) va vous générer votre MPD (Modèle Physique de Données)
  - des logiciels dédiés comme MySQL Workbench qui à partir d'un MLD (Modèle logique de données) va vous générer votre MPD.
  - Travailler à la main.

Notre MLD



- Analyse du modèle :
  - Une personne peut ne pas avoir de numéro de téléphone.
  - Deux personnes peuvent avoir le même nom et le même prénom (il faudrait sinon utiliser le nom et le prénom comme clef primaire).
  - Un même numéro de téléphone ne peut-être partagé entre plusieurs personnes.

Notre MPD

```
-- -----------------------------------------------------
-- Table `demo1db`.`personne`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `personne` (
  `idpersonne` INT NOT NULL AUTO_INCREMENT ,
  `nom` VARCHAR(45) NOT NULL ,
  `prenom` VARCHAR(45) NOT NULL ,
  PRIMARY KEY (`idpersonne`) )
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `demo1db`.`numero_telephone`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `numero_telephone` (
  `numero_telephone` VARCHAR(45) NOT NULL ,
  `personne_idpersonne` INT NOT NULL ,
  PRIMARY KEY (`numero_telephone`) ,
  CONSTRAINT `fk_numero_telephone_personne`
    FOREIGN KEY (`personne_idpersonne` )
    REFERENCES `personne` (`idpersonne` )
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

Peupler la base

- Nous pouvons aussi peupler notre base avec des insert

```
-- ----------------------------------------------------

-- Data for table `demo1db`.`personne`

-- ----------------------------------------------------

INSERT INTO `personne` (`idpersonne`, `nom`, `prenom`) VALUES (1, 'Durand', 'Jean');

INSERT INTO `personne` (`idpersonne`, `nom`, `prenom`) VALUES (2, 'Dupond', 'Paul');

-- ----------------------------------------------------

-- Data for table `demo1db`.`numero_telephone`

-- ----------------------------------------------------


INSERT INTO `numero_telephone` (`numero_telephone`, `personne_idpersonne`) VALUES ('400000000', 1);

INSERT INTO `numero_telephone` (`numero_telephone`, `personne_idpersonne`) VALUES ('600000000', 1);

INSERT INTO `numero_telephone` (`numero_telephone`, `personne_idpersonne`) VALUES ('100000000', 2);
```

- Il ne faut pas oublier de conserver le tout dans un script de déploiement

Configuration du SGBD

- Nous allons en développement utiliser le compte root (sans mot de passe)et créer la base demo1db.
- Pour accéder au SGBD vous pouvez soit utiliser :
  - Un client lourd
  - Une application Web (comme PHPMyAdmin)

Réalisation d'une page d'accueil

Le fichier servi par défaut est index.php

Nous allons l'utiliser pour afficher le contenu de la base :

1. Se connecter et choisir la base
2. Exécuter une requête et récupérer le résultat
3. Afficher le résultat
4. Se déconnecter

Réalisation d’une page d’accueil

```
<?php
    $db = mysql_connect('localhost', 'root', '');
    mysql_select_db('demo1db', $db);

 $sql = 'SELECT idpersonne,nom,prenom FROM personne';
    $req = mysql_query($sql) or die('Erreur SQL !<br>' . $sql . '<br>' .
   mysql_error());

    while ($data = mysql_fetch_assoc($req)) {
       print_r($data);
    }

    mysql_close();
    ?>
```

Réalisation d’une page d’accueil avec un peu de mise en forme

```
<?php while ($data = mysql_fetch_assoc($req)): ?>
<tr>
…
                    <td>
                        <?php echo $data['nom']; ?>
                    </td>
…
</tr>
<?php endwhile; ?>
```

Modifier la page d'accueil pour pouvoir modifier supprimer des personnes

Deux moyens principaux moyens en html pour envoyer de l'information à un serveur HTTP :

- Les formulaires
  - Envoi en POST ou GET vers une URL
  - Seul les input ou les select nommés sont transmis
- Les ancres
  - Envoi en GET

Formulaire d’ajout

```
<form action="ajouter.php" method="post">
        <p>
            Nom <input type="text" name="nom"/>
            Prénom <input type="text" name="prenom"/>
            <input type="submit" value="ajouter"/>
        </p>
    </form>
```

- Envoie la valeur de nom et prenom vers le script ajouter.php en mode post.

Script d’ajout

- Lire les données ($_GET[], $_POST[], $_REQUEST[])
- Effectuer une requête SQL
- Rediriger vers la page d’accueil (*header*("*Location*: … »))

Script d’ajout

```
$nom = $_POST['nom'];
 $prenom = $_POST['prenom'];

$db = mysql_connect('localhost', 'root', '');
mysql_select_db('demo1db', $db);

$sql='INSERT INTO personne(nom,prenom) VALUES
    (\''.$nom.'\',\''.$prenom.'\')';
$req = mysql_query($sql) or die('Erreur SQL !<br>' . $sql . '<br>' .
    mysql_error());
mysql_close();

header('Location: index.php');
```

Script de suppression

Il est invoqué via une ancre dont en GET mais repose sur le même principe que le script d'ajout.

Cependant l'url doit contenir le numéro de la personne à supprimer :

- En PHP : <a href="<?php echo './supprimer.php?id='.$data['idpersonne'];?>" > supprimer
- En HTML : supprimer

Script de suppression

```
$idpersonne = $_GET['id'];

$db = mysql_connect('localhost', 'root', '');
mysql_select_db('demo1db', $db);

$sql = 'DELETE FROM personne WHERE idpersonne=' .
    $idpersonne;
$req = mysql_query($sql) or die('Erreur SQL !<br>' . $sql .
    '<br>' . mysql_error());
mysql_close();

header('Location: index.php');
```

La modification et la consultation vont conduire à une nouvelle page

- Il faut recevoir l'identifiant de la personne
- Faire une requête pour récupérer les autres champs
- Faire une requête pour recouper les numéros de téléphone associés

La modification et la consultation vont conduire à une nouvelle page

```
$sql = 'SELECT idpersonne,nom,prenom FROM personne
    where idpersonne=' . $numero;
      $req = mysql_query($sql) or die('Erreur SQL !<br>' . $sql .
    '<br>' . mysql_error());


      $personne = mysql_fetch_assoc($req);
      $sql = 'SELECT numero_telephone FROM
    numero_telephone where personne_idpersonne=' .
    $numero;


      $req = mysql_query($sql) or die('Erreur SQL !<br>' . $sql .
    '<br>' . mysql_error());
```

Continuer le jeu

A chaque fois, il faut récupérer l'information du script ou de la page précédente en GET ou en GET, faire une ou des requêtes, afficher.

Rem : les inputs des formulaires de type hidden permettent d'envoyer de l'information sans l'afficher.

Pourquoi ne pas développer comme cela

Le déploiement

- Des URL en dur sont utilisées ce qui impose de parcourir le code pour les modifier.
- L'information de connexion est dans chaque page, ce qui impose aussi de parcourir les pages pour les modifier.

=>

- Utiliser un fichier contenant les constantes et l'inclure dans les pages

Pourquoi ne pas développer comme cela

La maintenance :

- Si le SGBD change, il faut tout reprendre et connaître les deux SGBD.

=>

- Utiliser une surcouche d'abstraction pour accéder au données (en TD PDO, en TP Doctrine).

Pourquoi ne pas développer comme cela

La maintenance :

- L'affichage, la navigation et l'accès aux données sont réalisés dans le même script ce qui impose de connaître les trois domaines et suivre les modifications de scripts en scripts

=>

- Utiliser le design pattern MVC

Pourquoi ne pas développer comme cela

La sécurité:

- Tous les scripts sont accessibles via une URL donc utilisables sans contrôle.
- Possibilité d'injecter du sql ou des scripts

=>

- Utiliser un framework