Interagir avec un SGBD

Sommaire

Sommaire

Historique des SGBD

- L'apparition des SGBD est fortement liée à l'évolution des moyens de stockage des données :
  - avant 1960 : utilisation de bandes magnétiques et de cartes perforées, traitement par lots (batch en anglais). Exemple : UNIVAC, prédit l'élection présidentielle de 1952.
  - avec l'apparition des premiers *disques durs* à tambour magnétique, on peut accéder *directement* aux données. Début de l'informatique transactionnelle. Exemple : IBM 305 (135 m^2, 1 t), 1956.
  - Rq : le premier PC d'IBM (le 5150, commercialisé en 1981) n'a pas de disque dur ! Seule la version ultérieure, le PC/XT sera doté d'un D.D. de . . .5 ou 10 Mo.
  - parallèlement, se développent des *systèmes de fichiers*, chaque fichier stockant lui-même des données sous forme d'*enregistrements*.
- Le premier SGBDR commercialisé est Oracle, vendu en 1979 par Relational Software. D'autres solutions sont ensuite commercialisées pour les grandes entreprises : DB/2 (IBM), Sybase (SAP).
- Avec le développement du web dans les années 1990, des alternatives moins onéreuses et accessibles aux PME et aux particuliers voient le jour : MySQL (devenu MariaDB en 95), PostgreSQL (96), SQLite.

Sommaire

Services rendus par un SGBD

Un SGBD doit assurer les services suivants :

- *persistance* et *durabilité* (en cas de panne) des données ;
- gestion des *accès concurrents* (car des actions vraiment simultanées pourraient mettre les données dans un état incohérent) ;
- *cohérence* à tout instant des données, en respectant les contraintes d'intégrité du modèle relationnel adopté ;
- *efficacité* du traitement des requêtes (selon le domaine, le temps moyen d'attente est variable) ;
- *sécurisation* des accès (on ne peut modifier les données sans y être autorisé).

Sommaire

Interactions avec un SGBD

- Un SGBD peut être utilisé :
  - par une application «autonome», développée dans un langage de haut-niveau : JDBC en Java, SQLite en Python par exemple ;
  - par une application web. Dans ce cas plusieurs solutions existent : ASP.NET avec C#, Ruby on Rails et le langage Ruby, Django et Python, Java et les JSP (Java Server Pages) (appelé JEE) et...PHP .
- PHP est l'alternative qui permet d'interfacer rapidement HTML et un SGBD . C'est un langage de programmation *exécuté côté serveur*, interprété, de haut niveau, impératif, orienté objet, open-source et gratuit, multiplateforme, faiblement typé. En 2018 on estimait qu'environ 80 % des sites web utilisaient PHP .
- PHP est doté d'extensions lui donnant la capacité de communiquer avec de nombreux SGBD : MySQL , PostgreSQL , Oracle,...Nous utiliserons ici l'extension PDO qui sait communiquer avec plusieurs SGBD , dont MySQL .
- On montre maintenant comment utiliser la combinaison PHP + MySQL .

Sommaire

Une (très) courte présentation de PHP

- Le code PHP est intégré au code HTML à l'aide des balises `<?php echo 'toto'; /* un commentaire */ ?>`.
- Le code PHP peut être placé n'importe où dans le code HTML : dans le corps, mais aussi dans l'en-tête, au milieu d'une balise.
- Les commentaires : multilignes avec `/* Ceci est un commentaire */`, de fin de ligne avec `// commentaire de fin de ligne`
- Les instructions se terminent par des points virgules.
- Les chaînes de caractères sont délimités par des guillemets simples ou doubles (cf. ci-dessous) et concaténées par un point.
- L'instruction `echo` insére le texte qui suit dans le code source HTML : `<?php echo "Ceci est du <strong>texte</strong>."; ?>`
- Les variables commencent par le caractère $: `<?php $x = 3; ?>`
- L'extension des fichiers contenant du code PHP est ".php".

Un exemple (voir ce tutoriel pour une introduction à PHP) :

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset = "utf-8">
5      <title>Mon premier programme en PHP.</title>
6    </head>
7    <body>
8      <ul>
9        <?php for($i=1; $i <= 10; $i++) {
10       ?>
11         <li>
12           <?php  echo 'Ceci est la ligne numéro'. $i; ?>
13         </li>
14       <?php
15       }
16       ?>
17     </ul>
18   </body>
19 </html>
```

Ex. 1 Tester dans un navigateur ce que réalise le code-source précédent (en ayant auparavant démarré le serveur lampp).

Sommaire

Transmettre des données de page en page

L'interaction d'une page web avec un utilisateur peut se faire par le biais d'un formulaire HTML dans lequel l'utilisateur «pilote» l'application web en rentrant ses choix.

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset = "utf-8">
5      <title>Formulaire de connexion.</title>
6    </head>
7    <body>
8      <p>Veuillez remplir les champs suivants:</p>
9      <form method="post" action="login.php">
10       <p>
11         <label for="pseudo">Entrez votre pseudo : </label>
12         <input type="text" name="pseudo" id="pseudo">
13         <br>
14         <label for="mdp">Entrez votre mot de passe : </label>
15         <input type="password" name="mdp" id="mdp">
16         <input type="submit" value="Valider">
17       </p>
18     </form>
19   </body>
20 </html>
```

```php
<!DOCTYPE html>
<html>
  <head>
    <meta charset = "utf-8">
    <title>Page de connexion</title>
  </head>
  <body>
    <?php
    if ($_POST['pseudo'] == 'toto' AND $_POST['mdp'] == 'mdp') {
      echo 'Bonjour '.$_POST['pseudo'].'!'; ?>
      <p> Ici la page où on accède seulement avec un pseudo et mot de
      passe corrects. </p>
    <?php
    }
    else {
      echo "Désolé, le pseudo et/ou le mdp sont incorrects."; ?>
      <p>cliquer <a href="formulaire.php">ici </a> pour revenir à la
        page de connexion.</p>
    <?php } ?>
  </body>
</html>
```

- L'attribut `method="post"` signifie que les données du formulaire (le pseudo et le mot de passe) seront transmises à la page cible sans être visibles dans la barre d'adresses du navigateur (contrairement au choix `method="get"`).
- L'attribut `action="login.php"` permet d'indiquer le nom de la page cible, qui va traiter les données du formulaire.
- Les attributs `name="pseudo"` et `name="mdp"` fixent les identificateurs des données qui seront traitées en PHP dans la page cible.
- L'attribut `type="text"` ou `type="password"` permet de fixer le type d'entrée qui sera saisie (texte, mot de passe, entier,...).
- Les attributs `id` dans les `input` permettent d'associer un identificateur à cette entrée. L'attribut `for` d'une balise `<label>` permet alors d'associer ce libellé à cette entrée, de sorte qu'un clic sur la zone du libellé sélectionne la zone de saisie qui lui est associée.
- L'envoi des données à la page cible se fait avec un bouton, qui est créé à l'aide d'une balise `input` ayant l'attribut `type="submit"`.

Ex. 2 Tester les deux pages précédentes et leur interaction. Vérifier que la connexion est réalisée avec le bon pseudo et le bon mot de passe, rejetée sinon.

- Malheureusement, cette façon d'envoyer des données est *non sécurisée*.
- Exemple 1 : l'utilisateur peut accèder[1] au code source du formulaire, modifier la méthode d'envoi des données (pour passer de `post` à `get`) et la page cible. Il voit alors transiter en clair les données (dont son mot de passe !) dans le contenu de l'url et peut les modifier.
- Exemple 2 : l'utilisateur peut également changer le nom des données échangées (renommer l'identificateur `pseudo` en `pseudo1` par exemple).
- Exemple 3 : l'utilisateur peut saisir du code Javascript dans une zone de saisie, par exemple un inoffensif `<script>alert('Coucou!');</script>`, qui peut parfois être exécuté. Avec plus de connaissances, il peut écrire un code Javascript qui récupère le contenu personnel des cookies (pseudo, mot de passe)...On parle de faille XSS (Cross-Site-Scripting).

1. Par clic droit, ou F12 pour accéder aux outils intégrés de développement.

- Sans nécessairement avoir forcément toujours des conséquences graves, de telles saisies peuvent provoquer un comportement non souhaité de l'application web, qui semble alors «buguer».

Ex. 3 Tester ce qui vient d'être dit précédemment : vérifier qu'avec la méthode `get` on voit les données envoyées par le formulaire apparaître dans l'url.

Pour remédier à ces problèmes on peut :

- Systématiquement vérifier, avec `isset`, que les données reçues correspondent bien à celle attendues et, dans certains cas, tester aussi leur type ;
- dans le cas de données pouvant conduire à l'exécution d'un code Javascript (par exemple quand une commande `echo` est utilisée), utiliser `htmlspecialchars` qui permet d'empêcher son exécution. On donne page suivante un exemple d'injection Javascript bloquée par cette fonction PHP .

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset = "utf-8">
5      <title>Formulaire de connexion.</title>
6    </head>
7    <body>
8      <p>Veuillez remplir les champs suivants:</p>
9      <form method="post" action="login.php">
10       <p>
11         <label for="nom">Entrez votre nom : </label>
12         <input type="text" name="nom" id="nom">
13         <input type="submit" value="Valider">
14       </p>
15     </form>
16   </body>
17 </html>
```

```php
<!DOCTYPE html>
<html>
  <head>
    <meta charset = "utf-8">
    <title>Page de connexion</title>
  </head>
  <body>
    <p>Bonjour <?php
                if (isset($_POST['nom'])) {
                  echo htmlspecialchars($_POST['nom']);
                }
                ?> !
    </p>
  </body>
</html>
```

Ex. 4 Vérifier qu'en enlevant `hthmlspecialchars` dans le code ci-dessus, on peut faire exécuter un code Javascript saisi dans la zone de texte (par exemple un simple `alert(...)` dans des balises <script>).

Sommaire

Connexion à MySQL

On suppose à partir de maintenant que la base de données `mediatheque` utilisée dans les chapitres précédents existe dans MySQL . Voici d'abord le code d'un formulaire pour rechercher des livres dans la médiathèque, puis (page suivante) la page qui réalise l'affichage des résultats obtenus (on donne seulement le corps, le reste étant inchangé par rapport aux exemples précédents) :

```
1  <body>
2    <p>Veuillez donner le nom de l'auteur dont vous cherchez les
3      livres dans la médiathèque:</p>
4    <form method="post" action="livres.php">
5      <p>
6        <label for="nom">Nom de l'auteur: </label>
7        <input type="text" name="nom" id="nom">
8        <input type="submit" value="Valider">
9      </p>
10   </form>
11 </body>
```

```php
<body>
  <?php
  try {
    $bdd = new PDO('mysql:host=localhost;dbname=mediatheque',
                   'root', '');
  }
  catch (Exception $e) {
    die('Erreur: ' . $e->getMessage());
  }
  $req = $bdd->prepare('SELECT a.nom, l.titre, l.annee FROM livre AS l
                        JOIN auteur_de AS ad ON ad.isbn = l.isbn
                        JOIN auteur AS a ON a.a_id = ad.a_id
                        WHERE a.nom = ?');
  $req->execute(array($_POST['nom']));
  while ($donnees = $req->fetch()) {
  ?>
    <p>
      <strong>auteur</strong>: <?php echo $donnees['nom']; ?><br>
      <strong>titre</strong>: <?php echo $donnees['titre']; ?><br>
      <strong>année</strong>: <?php echo $donnees['annee']; ?><br>
    </p>
  <?php
  }
  ?>
</body>
```

- La ligne 4 de la page précédente crée un objet de la classe PDO en appelant son constructeur (mot clef `new`). La classe PDO permet de créer une connexion avec de nombreuses bases de données (pas seulement avec MySQL).
- Dans cet appel, plusieurs arguments sont spécifiés :
  - `'mysql:host=localhost;dbname=mediatheque'` permet d'indiquer l'url du serveur de BdD (ici le poste lui-même) et le nom de la base de données à laquelle on veut se connecter ;
  - `'root'` est le nom d'utilisateur (ici LAMPP crée par défaut un accès à l'utilisateur `'root'`) ;
  - `''` est le mot de passe associé à cet utilisateur de la BdD (par défaut XAMPP crée un mot de passe vide pour l'utilisateur `'root'`).
- Le bloc `try` des lignes 3–6 d'une part et le bloc `catch` des lignes 7–9 d'autre part gèrent et interceptent une possible exception (erreur) levée au moment de la connexion [2]. En cas d'erreur, le programme est arrêté (`die`) avec affichage d'un message d'erreur (précédé de la chaîne `'Erreur: '`), concaténée à la chaîne obtenue par l'appel de la méthode `getMessage()` de l'objet `$e` (en PHP l'appel d'une méthode se fait avec `->`).

2. Qui peut se produire pour de mutiples raisons, utilisateur inconnu, mdp erroné, url du serveur inexacte,...