

Bases de données, ENS Cachan & Ulm

TP n° 6 – Requêtes préparées, Gestion de sessions

Pierre Senellart (pierre@senellart.com)

11 avril 2008

Le but de ce TP est de découvrir l'utilisation des requêtes préparées en JDBC et la gestion de sessions en PHP. Ce TP est assez court et requiert d'avoir terminé le TP n° 5, à l'exception de la toute dernière question.

1 Requêtes préparées (*prepared statements*) en JDBC

1. Les scripts PHP que vous avez écrits au TP n° 5 sont-ils robustes vis-à-vis des injections SQL ? Tester soigneusement et corriger au besoin.
2. Les problèmes d'injection de code viennent du mauvais couplage entre MySQL et PHP : une requête MySQL n'est rien d'autre, pour PHP, qu'une chaîne de caractères (construite par concaténation). JDBC propose des *requêtes préparées*, qui permettent une meilleure interaction entre Java et MySQL, et donc une meilleure sécurité. Un autre avantage des requêtes préparées est une efficacité accrue quand de multiples requêtes successives sont exécutées.

Une requête préparée est un objet de classe `PreparedStatement`, obtenu grâce à la méthode `prepareStatement` de l'objet connexion à la base de données. Cette méthode prend pour argument une chaîne de caractères contenant l'ordre SQL à exécuter, avec des caractères « ? » à la place des paramètres à substituer. On peut ensuite assigner une valeur à ces paramètres avec les méthodes `setString`, `setInt`, etc., de la requête préparée, qui prennent en argument le numéro du paramètre (à partir de 1, de gauche à droite, dans la requête préparée) et la valeur de ce paramètre. Se référer à la documentation Java de ces objets et méthodes pour plus d'informations.

Utiliser une requête préparée pour construire un programme Java affichant le solde d'un compte dont le numéro est passé en ligne de commande.

3. Une variante des requêtes préparées permet de faire des appels de procédure : c'est les objets de classe `CallableStatement`, obtenus grâce à la méthode `prepareCall` de l'objet connexion. Un exemple complet d'appel d'une procédure `foo` de type `(IN x INT, OUT y VARCHAR(50))` est :

```
CallableStatement s=conn.prepareCall("{ call foo(?,?) }");
s.setInt(1,42);
s.registerOutParameter(2, java.sql.Types.VARCHAR);
s.executeUpdate();
String y=s.getString(2);
```

Construire un programme Java permettant d'effectuer un virement en ligne de commande. Indiquer si le virement a été ou non effectué.

Remarque : PHP 5 propose également un mécanisme de requêtes préparées, mais pas par l'interface que nous avons utilisé jusqu'ici. Pour plus de renseignements, voir la documentation de l'interface `mysqli` sur <http://fr.php.net/mysqli>.

2 Gestion de sessions

L'application PHP actuellement réalisée permet à n'importe qui de faire des virements à partir de n'importe quel compte, ce qui n'est guère réaliste. Le but de cet exercice est d'ajouter un mécanisme de contrôle d'accès.

1. Adapter le schéma de base de données à cette nouvelle contrainte. Plutôt que de stocker des mots de passe en clair dans une table, on préférera toujours stocker un hachage cryptographique (par exemple, obtenu par la fonction MySQL `SHA` de ce mot de passe).
2. Étudier la partie *Authentication, Sessions* du cours disponible à l'URL <http://pierre.senellart.com/enseignement/2007-2008/ifips1web/cours8/cours8.pdf>.

3. Ajouter à l'application PHP un mécanisme d'authentification et de contrôle d'accès, en n'autorisant que les virements depuis un compte dont l'utilisateur est le propriétaire.
4. Tester soigneusement la sécurité de l'application réalisée.

3 Compléments

On souhaite maintenant ajouter la possibilité de demander un emprunt à la banque, à un certain taux d'intérêt, remboursable par versements à dates fixes. Imaginer les changements à effectuer, et programmer le tout. On pourra utiliser un programme Java tournant en arrière-plan pour simuler les remboursements. En cas de défaut de paiement, on pourra par exemple fermer le compte concerné et ajouter le client à une liste noire.

Faire en particulier attention à la sécurité et à la gestion des transactions.