PDO et MySQL

PDO (PHP Data Objets) :

PDO (*PHP Data Objects*) est une extension qui offre une couche d'abstraction de données introduite dans PHP 5, ce qui signifie qu'elle n'est pas liée à MySQL comme l'extension mysqli que nous avons abordée dans le TP précédent, mais indépendante de la base de données utilisée. Grâce à elle, le code devient portable très facilement, c'est-à-dire qu'elle modifie uniquement la ligne de connexion, d'une base de données MySQL à SQLite, PostgreSQL, Oracle et d'autres encore par exemple. PDO offre donc aussi bien l'avantage de la portabilité, de la facilité d'utilisation que de la rapidité.

L'extension PDO comprend les trois classes suivantes :

- La classe PDO, qui permet de créer des objets représentant la connexion à la base et qui dispose des méthodes permettant de réaliser les fonctions essentielles, à savoir l'envoi de requête, la création de requêtes préparées et la gestion des transactions.
- La classe PDOStatement, qui représente, par exemple, une requête préparée ou un résultat de requête SELECT. Ses méthodes permettent de gérer les requêtes préparées et de lire les résultats des requêtes.
- La classe PDOException qui permet de gérer et d'afficher des informations sur les erreurs à l'aide d'objets.

Les requêtes préparées :

Les requêtes préparées permettent de créer des requêtes SQL qui ne sont pas directement utilisables mais qui contiennent des paramètres auxquels on peut donner des valeurs différentes en fonction des besoins, par exemple, pour des appels répétitifs avec des valeurs différentes. Une requête préparée se présente, entre autres, sous la forme suivante :

SELECT prenom,nom FROM client WHERE ville=? AND id_client>=?

Dans laquelle les caractères **?** vont être remplacés par des valeurs quelconques en fonction des besoins du visiteur.

Ou encore avec des paramètres nommés :

SELECT prenom,nom FROM client WHERTE ville=:ville AND id_client=:id_client

Les paramètres nommés **:ville** et **:id_client**, dont les noms sont ici les mêmes que ceux des attributs de la table client, sont en fait arbitraires.

La démarche à suivre pour utiliser une requête préparée est la suivante :

1. Écrire la chaîne de requête comme paramètre de la méthode prepare() de l'objet PDO. Cette méthode retourne un objet de type PDOStatement qui représente la requête préparée.
2. Lier les paramètres dans l'ordre de leur apparition avec des valeurs ou des variables. Cette liaison peut se faire de plusieurs manières :

- En créant un tableau associatif de la forme :

$tab=array(':ville'=>$ville, 'id_client'=>$id_client) les variables $ville et $id_client ayant déjà des valeurs à ce stade. Ce tableau sera ensuite passé en paramètre à la méthode execute() détaillée ci-après.

- En appelant la méthode bindParam() des objets PDOStatement ou encore la méthode bindValue() pour chaque paramètre selon le modèle : bindParam(':ville',$ville,PDO::PARAM_STR) dans lequel le troisième paramètre désigne le type de la variable et peut prendre les valeurs PDO::PARAM_STR pour une chaîne et PDO::PARAM_INT pour un entier.

3. Exécuter la requête en appelant la méthode execute() de l'objet PDOStatement avec le tableau comme paramètre, s'il a été créé à l'étape 2, ou sinon sans paramètre.
4. Pour les requêtes préparées qui retournent des résultats, comme celles qui contiennent la commande SELECT, il faut ensuite lier les résultats à des variables PHP, avec la méthode bindColumn() selon les modèles :$regprep->bindColumn(1,$prenom) ou encore :

$regprep->bindColumn(prenom,$prenom)

Dans ce cas, la valeur de la colonne 1 désignée dans la requête sera contenue dans la variable $prenom, et ainsi de suite.

5. Les lignes de résultats sont obtenues en appliquant une des méthodes vues précédemment, comme fetch() ou fetchAll(), à l'objet PDOStatement représentant la

requête préparée, désignée ici par $regprep. Une ou plusieurs boucles, selon les cas, permet d'afficher les résultats en utilisant les variables créées à l'étape 4.

6. Libérer la mémoire occupée par l'objet PDOStatement avec la méthode close-Cursor().