Requêtes préparées et exécution

Préparation et exécution d'une requête

Une requête préparée est une requête qui est analysée, "compilée" et préparée par `MySQL`.

Elle présente plusieurs avantages :

Premièrement, une requête préparée peut **être exécutée plusieurs fois** avec des arguments identiques ou différents. Cela accélère les performances en diminuant les ressources nécessaires et en augmentant la vitesse des requêtes successives.

Deuxièmement, une requête préparée est nettoyée ce qui **empêche toute injection `SQL`**. C'est extrêmement important pour la sécurité de vos données.

Pour préparer une requête, il suffit d'utiliser la méthode `prepare()` des objets `PDO` et de lui passer la commande `SQL` à préparer.

Une fois la requête préparée, il suffit de l'exécuter avec la méthode `execute()`.

Voici un exemple :

```
<?php
$dsn = 'mysql:host=localhost;dbname=test';
$user = 'root';
$password = 'MOT DE PASSE';

try {
  $dbh = new PDO($dsn, $user, $password, [
    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION
  ]);
} catch (PDOException $e) {
  echo 'Connexion échouée : ' . $e->getMessage();
}

$stmt = $dbh->prepare(
  "INSERT INTO user VALUES
  (DEFAULT, 'adam', 'smith', 'adam@smith.fr', '123')"
);
$stmt->execute();
```

Utilisation de paramètres

Le plus souvent, vous n'exécuterez pas de commande contenant des valeurs fixes.

Les valeurs proviendront de votre code `PHP` et seront dynamiques.

Il faudra donc utiliser des **substitutions de paramètres**.

Nous allons voir ensemble toutes les possibilités offertes par `PDO` .

Substitution lors de l'exécution en utilisant des `?`

La manière la plus basique est d'utiliser des points d'interrogation, qui permettent d'indiquer que les valeurs seront passées dans le même ordre que la déclaration :

```
$stmt = $dbh->prepare("INSERT INTO user (firstname, lastname) VALUES (?,
?)");
$stmt->execute(['jacques', 'dupont']);
```

Substitution avant l'exécution en utilisant les méthodes `bindParam()` ou `bindValue()`

La méthode `bindValue()` permet de substituer un point d'interrogation ou un marqueur nommé par une valeur sur une requête préparée.

La méthode `bindParam()` permet la même chose, sauf qu'au lieu de substituer la valeur, elle lie le marqueur à une référence qui ne sera évaluée qu'à l'exécution.

```
$user = [
  'firstname' => 'jacques',
  'lastname' => 'dupont',
  'password' => 123
];
```

```
$stmt = $dbh->prepare("INSERT INTO user (firstname, lastname, password)
VALUES (?, ?, ?)");
$stmt->bindParam(1, $user['firstname']);
$stmt->bindParam(2, $user['lastname']);
$stmt->bindParam(3, $user['password']);
$stmt->execute();
```

Il est possible de préciser le type de la variable afin d'effectuer un transtypage (conversion forcée de type) :

```
$user = [
  'firstname' => 'jacques',
  'lastname' => 'dupont',
  'password' => 123
];


$stmt = $dbh->prepare("INSERT INTO user (firstname, lastname, password)
VALUES (?, ?, ?)");
$stmt->bindParam(1, $user['firstname'], PDO::PARAM_STR);
$stmt->bindParam(2, $user['lastname'], PDO::PARAM_STR);
$stmt->bindParam(3, $user['password'], PDO::PARAM_INT);
$stmt->execute();
```

Substitution avec des marqueurs nommés

Le plus souvent il est plus élégant d'utiliser des marqueurs nommés au lieu de points d'interrogation :

```
$user = [
  'firstname' => 'jacques',
  'lastname' => 'dupont',
  'password' => 123
];


$stmt = $dbh->prepare("INSERT INTO user (firstname, lastname, password)
VALUES (:firstname, :lastname, :password)");
$stmt->bindParam(':firstname', $user['firstname'], PDO::PARAM_STR);
$stmt->bindParam(':lastname', $user['lastname'], PDO::PARAM_STR);
$stmt->bindParam(':password', $user['password'], PDO::PARAM_INT);
$stmt->execute();
```

Vous pouvez dans ce cas également passer directement le tableau associatif si les clés correspondent exactement aux marqueurs :

```
$user = [
  'firstname' => 'jacques',
  'lastname' => 'dupont',
  'password' => 123
];


$stmt = $dbh->prepare("INSERT INTO user (firstname, lastname, password)
VALUES (:firstname, :lastname, :password)");
$stmt->execute($user);
```