SQL インジェクション攻撃に対抗するための「プレースホルダ」

例えば、以下のような SQL を実行する PHP プログラムがあったとします。

```
$sql = "SELECT * FROM user WHERE name='$name'";
```

この SQL は「user」というテーブルから、name が一致したデータを取り出す検索機能などでよく使われる SQL です。
末尾にある「$name」という PHP 変数は、ユーザーが検索ボックスに入力した値が入れられます。
例えば、検索ボックスに「東京太郎」と入力したら、実行される SQL は以下のようになります。

```
$sql = "SELECT * FROM user WHERE name='東京太郎'";
```

これが、「正常な SQL」です。
しかし、もし検索ボックスに「';DELETE FROM user--」と入力されたら、実行される SQL はこうなります。

```
$sql = "SELECT * FROM user WHERE name='';DELETE FROM user--";
```

冒頭の「';」によって、SELECT 文が終了されてしまい、続いて「DELETE FROM user」という新しい SQL が実行されてしまいます。末尾の「--」は SQL 上のコメントという意味なので、それ以降の文は無視されます。
つまり、開発者の全く想定しない「DELETE FROM user」(user テーブルの全データ削除)という命令が実行出来てしまうのです。
このように、ユーザーから入力された値を直接 SQL に結合してしまうと、万が一悪意のあるユーザーによって不正な値が入力された場合、想定外の SQL が実行されてしまう恐れがあります。このような攻撃手法を「SQL インジェクション」と言います。だから絶対に、SQL 文の中に直接 PHP の変数を書いてはいけません。
そこで登場するのが「プレースホルダ」という仕組みです。
プレースホルダは、上記のような SQL 文の中の「変動する箇所」に使用します。
プレースホルダを使って指定しておけば、その部分はあくまで「値」として処理され、万が一不正な値が入力されても、SQL 命令に関わるような「特殊文字」は無効化(エスケープ処理)されるため、SQL 文として実行されることはなくなります。SQL 文の中で「変動する箇所」には必ずプレースホルダを使いましょう。

プレースホルダの実際の使い方(ソース付き)

プレースホルダを使うための手順は 2 つです。
①SQL 文の変動箇所をプレースホルダ(:で始まる代替文字列)で指定する。
②bindValue で実際の値をプレースホルダにバインドする。
先ほどの SQL の例で説明すると、
①まず、このように変動箇所を「:で始まる代替文字列」で指定しておきます。

```
$sql = "SELECT * FROM user WHERE name=:name";
```

なお、型の判定もプレースホルダが行ってくれるので、文字列カラムの場合でもシングルクォーテーションは不要です。
プレースホルダの名前はカラム名と同じにしておくと分かりやすい。
②次に、プレースホルダに実際の値を「バインド」します。
これには「bindValue」というメソッドを使います。

```
bindValue(':name', $name, PDO::PARAM_STR);
```

書式)bindValue('プレースホルダ名', '実際にバインドするデータ', 'データの型');
第 3 引数の型の指定は、あらかじめ用意されている設定値から選択します。
よく使うのは以下のような型です。

【文字列型の場合】	PARAM_STR
【数値型の場合】	PARAM_INT
【ラージオブジェクト(画像データなど)の場合】	PARAM_LOB
【NULL の場合】	PARAM_NULL