Biblioteki mysqli obsługuje instrukcje przygotowane. Są one doskonałym narzędziem skracania czasu pracy w sytuacji, gdy wykonywana jest znaczna liczba zapytań z różnymi zestawami danych. Instrukcje te chronią także przed punktowymi atakami SQL.

Idea instrukcji składowanej polega na tym, że do serwera MySQL wysyłany jest szablon zapytania, które będzie wykonywane, a następnie oddzielnie wysyła się dane. Do jednej z instrukcji przygotowywanej można wysyłać wiele zestawów takich samych danych, co okazuje się szczególnie użyteczne w przypadku masowego wstawiania danych.

Instrukcje przygotowywane moglibyśmy wykorzystać w skrypcie wstaw_ksiazke.php w następujący sposób:

{codecitation class='brush: php'}
$zapytanie = "insert into ksiazki values (?, ?, ?, ?)";
$instrukcja = $db->prepare($zapytanie);
$instrukcja -> bind_param(„sssd”, $isbn, $autor, $tytul, $cena);
$instrukcja->execute();
echo $instrukcja->affected_rows. ‘książka zapisana do bazy.’;
Instrukcja->close();

{/codecitation}

Przeanalizujmy ten kod wiersz po wierszu.

Gdy składane jest zapytanie, zamiast wstawiania zmiennych, jak miało to miejsce dotychczas, w miejsce każdego elementu danych wpisywany jest znak zapytania. Wokół znaków zapytania nie powinno się umieszczać żadnych cudzysłowów ani innych separatorów.

W drugim wierszu następuje wywołanie *$db -> prepare()*, które w wersji proceduralnej ma postać *mysqli_stmt_prepare().*
Wiersz ten tworzy obiekt instrukcji lub zasobu, który następnie zostanie użyty w trakcie rzeczywistego przetwarzania.

Obiekt instrukcji posiada metodę o nazwie *bing_param()*. Zadaniem *bind_param()* jest wskazanie PHP zmiennych, które powinny zastąpić znaki zapytania. Pierwszy parametr jest ciągiem formatującym, podobnym do ciągu formatującego używanego w funkcji *printf()*
. Wartość przekazywana w naszym przykładzie („sssd") oznacza, że cztery kolejne parametry są odpowiednio ciągiem znaków, ciągiem znaków, ciągiem znaków i liczbą typu double. Innymi dostępnymi znakami są 'i' oznaczające liczbę całkowitą integer oraz b wskazujące typ blob. Po tym parametrze należy wymienić zmienne w liczbie równej liczbie znaków zapytania umieszczonych w instrukcji. W takiej właśnie kolejności znaki zapytania ulegną zamianie.

Wywołanie *$instrukcja ->execute()* (w wersji proceduralnej *mysqli_stmt_execute()*) uruchamia wykonanie zapytania. W jego następstwie można odczytać liczbę wierszy wynikowych oraz zamknąć instrukcję.

Na czym zatem polega przydatność instrukcji przygotowywanych? Ciekawym rozwiązaniem jest to, że można zmienić wartości czterech zmiennych dowiązanych i na nowo wykonać instrukcję bez konieczności jej ponownego przygotowania. Mechanizm taki jest przydatny w trakcie wykonywania w pętli masowych operacji wstawiania danych.
Tak samo jak parametry, wiązać można również wyniki. Względem zapytań typu *SELECT* można wywołać polecenie
$instrukcja->bind_result()
(lub
mysqli_stmt_bing_result()
) w celu pobrania listy zmiennych, którymi powinny zostać wypełnione kolumny wynikowe. Za każdym razem, gdy wywoływana będzie funkcja
$instrukcja->fetch()
(lub
mysqli_stmt_fetch()
), wartości kolumn z każdego wiersza zestawu wyników wypełnią dowiązane zmienne. Np., w skrypcie wyszukującym książki, który przedstawiliśmy wcześniej, można by wywołać polecenie

{codecitation class='brush: php'}
$instrukcja ->bing_result($isbn, $autor, $tytul, $cena);{/codecitation}
W celu dowiązania tych czterech zmiennych do czterech kolumn, które zostaną zwrócone przez zapytanie. Po wykonaniu instrukcji

{codecitation class='brush: php'}
$instrukcja->execute();

{/codecitation}
Można w pętli wywoływać polecenie

{codecitation class='brush: php'}
$instrukcja->fetch();{/codecitation}

Wówczas w wyniku każdego takiego wywołania pobierany jest kolejny wiersz wynikowy i następuje jego wczytanie do czterech zmiennych dowiązanych.
W tym samym skrypcie można również użyć funkcji *mysqli_stmt_bind_param()* oraz *mysqli_stmt_bing_result().*

Źródło: Vademecum Profesjonalisty - wyd. Helion.