Курс:

«Создание web–приложений, исполняемых на стороне сервера при помощи языка программирования PHP, СУБД MySQL и технологии Ajax»

Модуль 02

ТЕМА: ЗАПРОСЫ К БАЗЕ ДАННЫХ

Работа с MySQL в PHP

Специальные встроенные инструменты для работы с MySQL позволяют просто и эффективно работать с этой СУБД: выполнять любые запросы, читать и записывать данные, обрабатывать ошибки.

Сценарий, который подключается к базе данных, выполняет запрос и показывает результат, будет состоять всего из нескольких строк. Для работы с MySQL не нужно производить дополнительную установку или же настройку библиотек.

Для работы с базой данных MySQL используются следующие расширения:

1. **MySQL Improved (MySQLi)** — расширение для доступа к базе данных MySQL. Имеет как процедурный, так и объектно-ориентированный интерфейсы взаимодействия.
2. **PHP Data Objects (PDO)** — расширение, предоставляющее универ-сальный интерфейс для доступа к различным базам данных. PDO предлагает единые методы для работы с различными

базами данных, хотя текст запросов может немного отличаться из-за различий в используемых системах управления базами данных.

Сравнительная таблица

Наименование	PDO	MySQLi
Поддержка баз данных	12 различных драйверов, 18 систем управления базами данных	Только MySQL
Программный интерфейс доступа	Только объектно-ориентированный	Объектно-ориентированный, процедурный
Именованные параметры	Да	—
Объектное отображение	Да	Да
Подготовленные выражения (сторона клиента)	Да	—
Хранимые процедуры	Да	Да

Работа с базой данных разделяется на следующие этапы:

1. Установить подключение к серверу базы данных, передав необходимые параметры: адрес, логин, пароль и другие.
2. Убедиться, что подключение прошло успешно: сервер базы данных доступен, а все данные и параметры переданы верно.
3. Сформировать правильный SQL запрос.
4. Убедиться, что запрос был выполнен успешно.
5. Получить результат от базы данных в виде массива из записей.
6. Использовать полученные записи в своём сценарии.

Подключение к базе данных MySQL при помощи PHP Data Objects

Для установки подключения к базе данных необходимо первоначально объявить следующие переменные:

1. Адрес сервера (по умолчанию localhost)
2. Наименование базы данных.
3. Порт подключения к базе данных (по умолчанию 3306)
4. Кодировка.
5. Имя пользователя.
6. Пароль пользователя.

Далее требуется объявить переменную, в которую будут передаваться все необходимые параметры. Полный синтаксис выглядит следующим образом:

```php
<?php

define("HOST", "localhost");
define("DATABASE", "sep221");
define("PORT", 3306);
define("CHARSET", "utf8mb4");
define("USER", "root");
define("PASSWORD", "");

$pdo = new PDO(
    "mysql:host=" . HOST . ";" .
    "port=" . PORT . ";" .
    "dbname=" . DATABASE . ";" .
    "charset=" . CHARSET,
    USER, PASSWORD
);
```

В переменной `$pdo` происходит объявление экземпляра класса PDO для работы с базой данных. Данный класс имеет определённые методы для произведения дальнейших операций внутри базы данных.

Возможно использования конструкции обработчика ошибок на случай, если подключение к базе данных отсутствует. Однако в этом нет необходимости, поскольку PHP в любом случае покажет ошибку, что делает подобных код избыточным.

Также переменную можно сделать глобальной и вынести содержимое подключения в отдельный файл. Тогда использование базы данных в других файлах будет выглядеть так:

```
global $pdo;
require_once("db.php");
```

Выборка значений из таблицы

Выборка из таблицы происходит при помощи объявления переменной для задания SQL-запроса, после чего результаты выполнения требуется определить в новой переменной (например, «результат» — `$res`), а после — передать экземпляру класса `PDO` параметры запроса для метода `query()`. Полный синтаксис выглядит следующим образом:

```
$sql = "SELECT * FROM <таблица>";
$res = $pdo->query($sql);
```

Если же в запрос передаётся хотя бы одна переменная, то этот запрос в обязательном порядке должен выполняться только через подготовленные выражения — обычный SQL-запрос, в котором вместо переменной ставится специальный маркер. PDO поддерживает позиционные маркеры, для кото-рых важен порядок передаваемых переменных, и именованные, для которых порядок не важен. Примеры:

```
$sql = "SELECT name FROM users WHERE b_day = ?";
$sql = "SELECT name FROM users WHERE b_day =
:b_day";
```

Чтобы выполнить такой запрос, сначала его надо подготовить с помощью метода `prepare()`. Он возвращает *PDO statement* — подготовленный запрос к базе данных, но без данных. Чтобы их получить, надо выполнить запрос, предварительно передав в него переменные одним из способов:

1. Чаще всего можно просто выполнить метод `execute()`, передав ему массив с переменными:

```
$res = $pdo->prepare("SELECT name FROM users
WHERE b_day = ?");
$res->execute([$b_day]);
```

2. В случае именованных маркеров в `execute()` должен передаваться массив, в котором ключи должны совпадать с именами маркеров:

```
$res = $pdo->prepare("SELECT name FROM users
WHERE b_day = :b_day");
$res->execute(["b_day" => $b_day]);
```

3. Последним шагом станет преобразование полученных данных в ассоциативный массив и записи данного результата в переменную при помощи следующих команд:

```
$users = $res->fetchAll(PDO::FETCH_ASSOC);
```

После этого можно использовать PDO statement для вывода. Например, с использованием цикла перебора массивов:

```
foreach ($users as $user) {
    echo $user->name . "<br>";
}
```