Работа с базой PHP MYSQL

Для работы с Mysql выбирать можно из трёх API:

- старое доброе расширение mysqli. - процедурный стиль.

Объектно-ориентированный стиль

- mysqli. Эта библиотека **не предназначена** для испрользования напрямую в коде. А только как строительный материал для создания библиотеки более высокого уровня.
- PDO - PHP Data Objects. Этот класс, сокращенно именуемый PDO, предоставляет методы для работы с объектами.

	PDO	MySQLi
Поддержка баз данных	12 различных драйверов	Только MySQL
API	ООП	ООП+процедурная часть
Соединение	Просто	Просто
Именованные параметры	Есть	Нет
Объектное отображение	Есть	Есть
Подготовленные выражения (сторона клиента)	Есть	Нет
Производительность	Хорошая	Хорошая
Хранимые процедуры	Есть	Есть

Ключевым преимуществом PDO перед MySQLi является его могучая поддержка различных баз данных. На момент написания **PDO может использовать 12 драйверов.** А MySQLi - поддерживает только **MySQL**.

Соединение с сервером и базой (pdo)

```
$dsn = "mysql:host=$host;dbname=$db;charset=$charset";
    $opt = array(
        PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
        PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC
    );
    $pdo = new PDO($dsn, $user, $pass, $opt);
```

PDO::FETCH_ASSOC (integer)Указывает, что метод, осуществляющий выборку данных, должен возвращать каждую строку результирующего набора в виде ассоциативного массива

PDO::ERRMODE_EXCEPTION (integer)Предписание выбрасывать исключение PDOException в случае ошибки

http://fi2.php.net/manual/ru/pdo.constants.php

- **PDO::FETCH_ASSOC:** возвращает массив с названиями столбцов в виде ключей
- **PDO::FETCH_BOTH (по умолчанию):** возвращает массив с индексами как в виде названий стобцов, так и их порядковых номеров
- **PDO::FETCH_BOUND:** присваивает значения столбцов соответствующим переменным, заданным с помощью метода ->bindColumn()
- **PDO::FETCH_CLASS:** присваивает значения столбцов соответствующим свойствам указанного класса. Если для какого-то столбца свойства нет, оно будет создано
- **PDO::FETCH_INTO:** обновляет существующий экземпляр указанного класса
- **PDO::FETCH_LAZY:** объединяет в себе PDO::FETCH_BOTH и PDO::FETCH_OBJ
- **PDO::FETCH_NUM:** возвращает массив с ключами в виде порядковых номеров столбцов
- **PDO::FETCH_OBJ:** возвращает анонимный объект со свойствами, соответствующими именам столбцов
- На практике вам обычно хватит трех: FETCH_ASSOC, FETCH_CLASS, и FETCH_OBJ. Чтобы задать формат данных, используется следующий синтаксис:

Обработка ошибок

- PDO предлагает на выбор 3 стратегии обработки ошибок:
- PDO::ERRMODE_SILENT
- Это режим по умолчанию. PDO просто предоставит вам код ошибки
- PDO::ERRMODE_WARNING
- Помимо задания кода ошибки PDO выдаст обычное E_WARNING сообщение.
- PDO::ERRMODE_EXCEPTION
- Помимо задания кода ошибки PDO будет выбрасывать исключение PDOException, свойства которого будут отражать код ошибки и ее описание.

PDO константы атрибуты

- http://php-zametki.ru/php-prodvinutym/58-pdo-konstanty-atributy.html

- PDO::ATTR_DEFAULT_FETCH_MODE
  Доступный начиная с PHP 5.2.0
- Задаёт тип получаемого результата по-умолчанию:

Так делать не надо при обработке ошибок

```
try {
    $dbh = new PDO($dsn, $user, $password);
  }
catch (PDOException $e) {
    die('Подключение не удалось: ' . $e->get
  Message());
  }
```

Пример

```php
<?php
$host="localhost";
$db="store";
$charset="utf8";
$user="root";
$pass="";
$dsn = "mysql:host=localhost;dbname=store";
$opt = array(
    PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC
);

$pdo = new PDO($dsn, $user, $pass,$opt);
?>
```

fetch() - является аналогом функции mysq_fetch_array() .

посоветую применять fetch() в режиме FETCH_LAZY:

В этом режиме не тратится лишняя память, и к тому же к колонкам можно обращаться любым из трех способов - через индекс, имя, или свойство.

```
$result = $mysqli->query("select * from customers");
while($row = $result->fetch(PDO::FETCH_LAZY))
                                                    {
echo $row['cname'] . "\n";
echo $row[0] . "\n";
      echo $row->cname . "\n";
       }
```

Выполнение запросов.

Для выполнения запросов можно пользоваться двумя методами.

Если в запрос не передаются никакие переменные, то можно воспользоваться функцией query().

```
$stmt = $pdo->query('SELECT name FROM users');
while ($row = $stmt->fetch())
{
    echo $row['name'] . "\n";
}
```

Выборка данных PDO

```
$result = $pdo->query("select * from customers");
                while($row = $result->fetch(PDO::FETCH_LAZY))
                                                {
                    echo $row['cname'] . "\n";
                     echo $row[0] . "\n";
echo $row->cname . "\n";
                    }
```

Подготовленные выражения

Если же в запрос передаётся хотя бы одна переменная, то этот запрос в обязательном порядке должен выполняться только через **подготовленные выражения.**

Это обычный SQL запрос, в котором вместо переменной ставится специальный маркер - плейсхолдер. PDO поддерживает позиционные плейсхолдеры (?), для которых важен порядок передаваемых переменных, и именованные (:name), для которых порядок не важен.

```
$sql = 'SELECT name FROM users WHERE email = ?';
$sql = 'SELECT name FROM users WHERE email = :email';
```

Чтобы выполнить такой запрос, сначала его надо подготовить с помощью функции prepare().

```
$stmt = prepare('SELECT name FROM users WHERE email = ?')
$stmt->execute(array($email));
```

```
$stmt = prepare('SELECT name FROM users WHERE email = :email');
$stmt->execute(array('email' => $email));
```

Как видно, в случае именованных плейсхолдеров в execute() должен передаваться массив, в котором ключи должны совпадать с именами плейсхолдеров.

```
$stmt = $pdo->prepare('SELECT name FROM users W
HERE email = ?');
$stmt->execute([$_GET['email']]);
foreach ($stmt as $row)
{
    echo $row['name'] . "\n";
}
```

ВАЖНО: Подготовленные выражения - основная причина использовать PDO, поскольку это **единственный безопасный способ** выполнения SQL запросов, в которых участвуют переменные.

Создаем форму

```
<form name="add_c" action="<?php echo $_POST['PHP_SELF']; ?>" method="post">
<input type="text" name="cnum" placeholder="Ввеіть код" required/>
<input type="text" name="cname" placeholder="Ввеіть назву" required/>
<button id='ins' name='insert' type='submit'>Вставить</button>
</form>
```

Вставка данных в таблицу PDO

```php
if(isset($_POST['insert'])){
    $cnum = $_POST['cnum'];
    $sql = "insert into customers(cnum,cname) values('$cnum','".$_POST['cname']."')";
    $stm = $pdo->prepare($sql);
    $stm->execute($values);
}
```

PDO::prepare — Подготавливает запрос к выполнению и возвращает ассоциированный с этим запросом объект

PDOStatement::execute — Запускает подготовленный запрос на выполнение

Добавляем анализатор действия на удаление PDO

```
        if(isset($_POST['del'])){
            $sql = "delete from customers where cnum='".$_POST['cnum']."'";
$stm = $pdo->prepare($sql);
$stm->execute($values);
echo '<script type="text/javascript">
location="http://localhost/kyrsl/php_1/php_mysql.php";
</script>';
        }
```

Редактирование данных PDO

Добавляем кнопку редактирования

```
printf("<form name='change' action='%s' method='post'>
    <input type='hidden' name='cnum' value='%s'/>
    <button id='edit' name='change' type='submit'>Редактировать</button>
</form>",$_POST['PHP_SELF'],$row['cnum']);
```

Добавляем форму для редактирования полей

1. При загрузке формы первоначально загружается форма добавления данных.
2. После нажатия кнопки «Редактировать» форма меняется на форму редактировать.
3. Пишем код проверки была ли нажата кнопка «Редактировать».

```
if(isset($_POST['change']))
            {
$result = $pdo->query("select * from customers where cnum='".$_POST['cnum']."'");
                $row = $result->fetch();
                ?>
                <h2>Зміна запису</h2>
            <form name="change" method="post" action="<?php echo $_POST['PHP_SELF'];?>">
                <table>
                    <input type="hidden" value="<?php echo $row['cnum']; ?>" name="cnum" />
                    <tr><th>Назва:</th><td><input type="text" name="cname" value="<?php echo $row['cname'];?>" required/></td></tr>
                    <tr><th></th><td><input type="submit" value="Змінити" name="change_q"/></td></tr>
                </table>
            </form>
                <?php

            }
else        {?>
                    <form name="add_c" action="<?php echo $_POST['PHP_SELF']; ?>" method="post">
                    <input type="text" name="cnum" placeholder="Ввеіть код" required/>
                    <input type="text" name="cname" placeholder="Ввеіть назву" required/>
                    <button id='ins' name='insert' type='submit'>Вставить</button>
                    </form>
    <?php }?>
```

Обработчик нажатия кнопки редактировать PDO

```
if(isset($_POST['change_q'])){
$sql="update customers set cname='".$_POST['cname']."' where cnum='".$_POST['cnum']."'";
$stm = $pdo->prepare($sql);
$stm->execute($values);
echo '<script type="text/javascript">
location="http://localhost/kyrs1/php_1/php_mysql.php";
</script>';
}
```

Задание

Сделать все что делали для оставшихся 2 таблиц.

Orders and salespeople