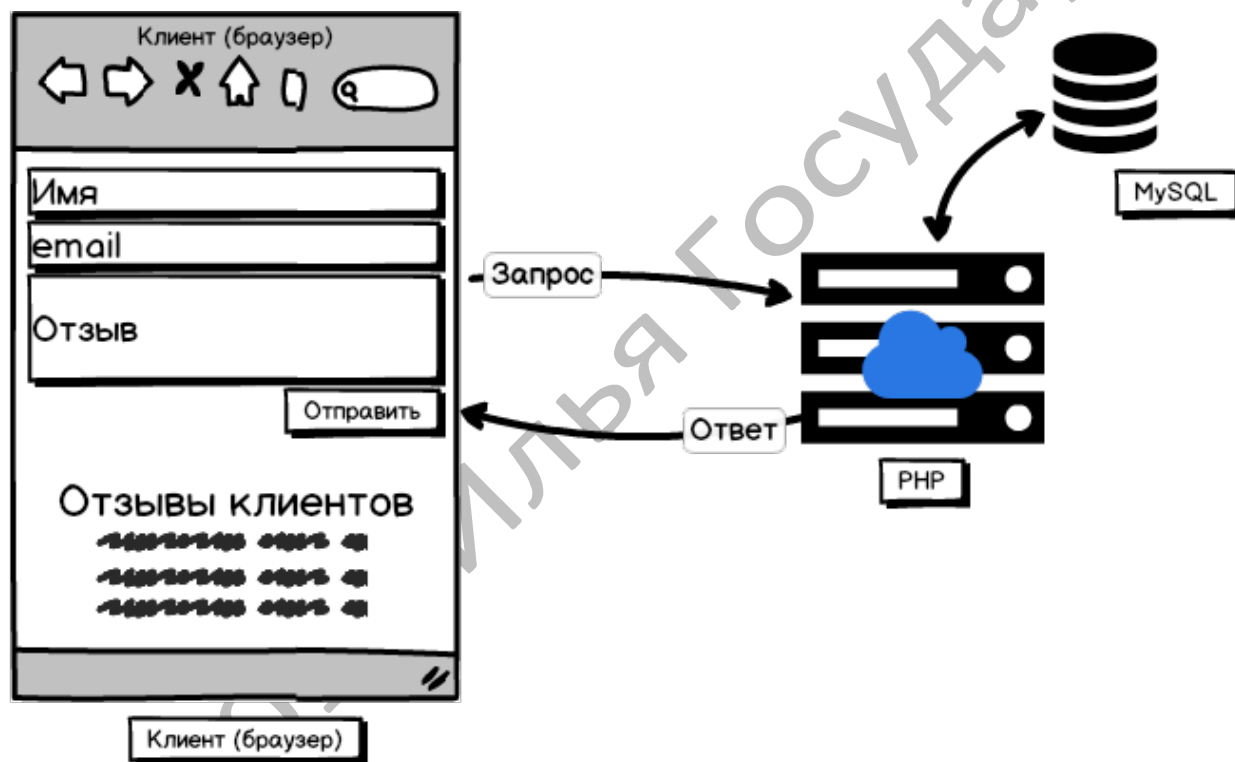


Создадим приложение, которое демонстрирует базовые возможности связки PHP + MySQL.

Приложение представляет собой простейшую гостевую книгу или журнал отзывов. Отзыв - это сущность данных, состоящая из трёх полей: имя, электронная почта и собственно текст отзыва, подобный твиту.

Составные части нашего приложения:

1. Клиентская часть - форма с полями, куда посетитель сайта вводит три части своего отзыва, и список существующих отзывов.
2. Серверная часть:
 - 2а. Сценарий на PHP, который принимает отзыв и выдаёт из базы данных существующие отзывы.
 - 2б. База данных MySQL, которая хранит отзывы.



В простом приложении не будет аутентификации/авторизации, т.е. отзывы оставляет любой посетитель и нельзя достоверно найти отзывы, принадлежащие одному и тому же посетителю - каждый может зайти на форму и отослать несколько отзывов от разных имён.

Спроектируем структуру данных. Она будет состоять из четырёх полей - три уже рассмотрены выше, а четвёртая - id - уникальный номер; он будет присваиваться в порядке добавления каждому новому отзыву и использоваться для внутренних целей приложения.

У нас будет база данных с одной таблицей reviews (отзывы), состоящей из записей, каждая из которых содержит эти четыре поля.

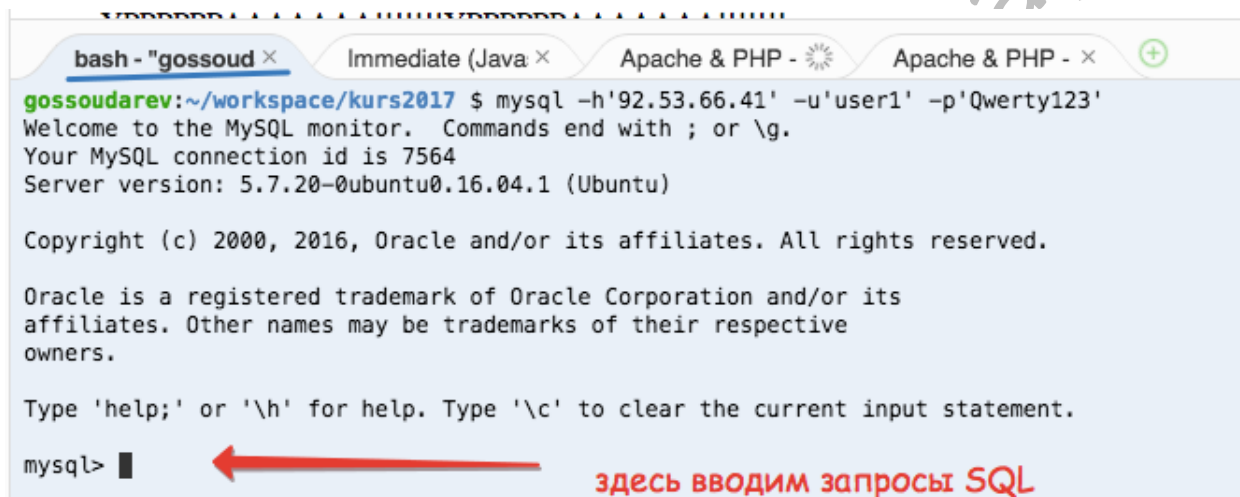
Пример приложения с аналогичной структурой:
<http://itspbappo.ru/elective/#>

Нам нужно иметь возможность свободно подключаться к БД в учебных целях с других серверов, в первую очередь с c9.io - это можно обеспечить, создав виртуальный сервер с приложением LAMP, например, на vscale.io - предположим, что у нас есть такой сервер, доступный по IP-адресу 92.53.66.41 и на нём создана учебная БД user1 и пользователь с тем же именем, которому разрешено подключаться извне.

В среде c9.io есть встроенная командная оболочка. Введём там команду подключения:

```
mysql -h'92.53.66.41' -u'user1' -p'Qwerty123'
```

(конкретные параметры этой команды, указанные в одинарных кавычках могут быть другими)



```
bash - "gossoud" x Immediate (Java) x Apache & PHP - Apache & PHP - x
gossoudarev:~/workspace/kurs2017 $ mysql -h'92.53.66.41' -u'user1' -p'Qwerty123'
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 7564
Server version: 5.7.20-0ubuntu0.16.04.1 (Ubuntu)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> 
```

здесь вводим запросы SQL

Посмотрим список существующих БД:

```
mysql> show databases;
```

```
+-----+
| Database |
+-----+
| information_schema |
| user1 |
+-----+
2 rows in set (0.27 sec)
```

Подключимся к нашей БД и выведем список таблиц (это будет одна таблица, заранее подготовленная; команда для создания таблицы рассмотрена ниже):

```
mysql> use user1;
```

Database changed

```
mysql> show tables;
```

Empty set (0.00 sec)

```
mysql>
```

```

+-----+
| Tables_in_user1 |
+-----+
| reviews        |
+-----+

```

1 row in set (0.04 sec)

Вот

Вот команды для работы с таблицей:

Создание таблицы
CREATE TABLE `reviews` (`id` INT NOT NULL AUTO_INCREMENT, `name` TEXT NOT NULL, `email` TEXT NOT NULL, `text` TEXT NOT NULL, PRIMARY KEY (`id`)) ENGINE = INNODB CHARACTER SET utf8 COLLATE utf8_unicode_ci;
Удаление таблицы
DROP TABLE `reviews`;
Получение её структуры
<pre> SELECT `COLUMN_NAME` from `INFORMATION_SCHEMA`.`COLUMNS` WHERE `TABLE_NAME`='reviews'; +-----+ COLUMN_NAME +-----+ id name email text +-----+ 4 rows in set (2.17 sec) </pre>

===

Если таблицу только что создали, в ней нет ничего

```
mysql> SELECT * FROM `reviews` WHERE 1;
Empty set (0.01 sec)
```

Иначе выводится список строк

mysql> SELECT * FROM `reviews` WHERE 1;			
id	name	email	text
1	elias	elias.b.goss@gmail.com	Очень хороший сайт!
2	Petr Ivanov	user@yandex.ru	Шикарно!
3	Sasha	sasha@mail.ru	Мне понравился сайт, но мало данных
4	Анонимус	guy.fawkes@protonmail.com	Несекурно, взломать пара пустяков
5	light	light@dark.net	Тут светло!
6	little me	lopez@valdo.ch	Надо улучшать интерфейс
7	loner	loner1@lone.com	Ведётся большая работа над сайтом
8	Сергей Брин	dev@developers.usa.net	Хотел бы стать участником команды
9	Bill Gates	pres@macrosoft.org	Готов проспонсировать ваш сайт
10	Ilya Gosudarev	ilia.gossoudarev@gmail.com	Присоединяюсь к предыдущим отзывам
10 rows in set (0.04 sec)			

Так выглядят команды для вставки значений:

```
INSERT INTO `reviews` (`name`, `email`, `text`) VALUES ('elias','elias.b.goss@gmail.com', 'Очень хороший сайт!');
```

```
INSERT INTO `reviews` (`name`, `email`, `text`) VALUES ('Petr Ivanov','user@yandex.ru', 'Неплохо!');
```

Мы можем конкретно указать значение какого поля и какой записи (по id) мы хотим получить:

```
mysql> SELECT `text` FROM `reviews` WHERE `id`='2';
```

```
+-----+
| text          |
+-----+
| Неплохо!      |
+-----+
1 row in set (0.00 sec)
```

```
mysql> UPDATE `reviews` SET `text`='Шикарно!' WHERE `id`='2';
```

Query OK, 1 row affected (0.00 sec)

Rows matched: 1 Changed: 1 Warnings: 0

```
mysql> SELECT `text` FROM `reviews` WHERE `id`='2';
```

```
+-----+
| text          |
+-----+
| Шикарно!      |
+-----+
1 row in set (0.00 sec)
```

db36.valuehost.ru > gossouda_guest > reviews "InnoDB free: 10231808 kB"

Showing rows 0 - 1 (~2¹ total, Query took 0.0023 sec)

```
SELECT *
FROM `reviews`
LIMIT 0, 30
```

Show: 30 row(s) starting from record # 0

in horizontal mode and repeat headers after 100 cells

Sort by key: None

+ Options

	id	name	email	text
<input type="checkbox"/>	1	elias	elias.b.goss@gmail.com	Очень хороший сайт!
<input type="checkbox"/>	2	Petr Ivanov	user@yandex.ru	Шикарно!

Check All / Uncheck All With selected:

Вид того же самого под управлением phpMyAdmin

Нужно обратить внимание на то, что при вставке значений с помощью INSERT мы не указывали значения ID. Они встали сами благодаря **auto_increment**

db36.valuehost.ru > gossouda_guest > reviews

Field	Type	A.I.	h
id	INT	<input checked="" type="checkbox"/>	

Самостоятельно выясните, какой запрос позволяет удалять данные.

Аббревиатура CRUD означает типичный набор операций с БД:

C(reate) - создать / вставить значения

R(read) - прочитать / выбрать значения

U(pdate) - обновить / изменить значения

D(elete) - стереть / удалить значения

Естественно, реалистичный кейс использования запросов - это их автоматическое выполнение в сценариях. При этом следует учитывать особенности сосуществования синтаксисов SQL и PHP.

Мы видим, что в SQL используются кавычки двух видов: для имён полей, таблиц и баз данных, а также для значений. Запросы совершенно естественно отличаются друг от друга и этими именами, и значениями. Как встраивать их в синтаксис PHP?

Можно заключать запрос в двойные кавычки, а его сменные части выносить за пределы запроса в отдельные переменные. Существует и вариант параметризации запроса, который превращает конкретный запрос в своего рода шаблон. Об этом чуть ниже.

Сначала рассмотрим простой способ обратиться к БД из PHP. Выделены строки, создающие подключение. PDO - это механизм, позволяющий обращаться с данными в объектной форме (с оператором ->)

```
<?php
header('Content-type: text/html; charset=utf-8 ');
echo '<h1>Извлечение данных...</h1>';
$dbloc = '92.53.66.41';
$dbuser = $dbname = 'user1';
$dbpass = 'Qwerty123';
$dsn = "mysql:host={$dbloc};dbname={$dbname}";
$conn = new PDO($dsn, $dbuser, $dbpass);
$conn->exec('SET CHARACTER SET utf8');

$id = 4;
$sql = "SELECT `text` FROM `reviews` WHERE `id` = '{$id}'";
$result = $conn->query($sql);
echo '<i>' . $result->fetch(PDO::FETCH_OBJ)->text . '</i>';
```

Здесь мы извлекаем значение поля `text` из той записи в таблице `reviews`, в которой значение поля `id` равно '4', при этом значение не указывается напрямую в запросе, а вместо него туда встроена переменная id. Встраивание переменной возможно благодаря использованию двойных кавычек вокруг запроса и фигурных скобок вокруг имени переменной. Сравните это с template strings в JavaScript, где знак доллара являлся бы частью синтаксиса строки и находился бы перед фигурными скобками (было бы не {\$id} а \${id}) и кавычки были бы одинарными обратными косыми.

Мы можем перефразировать этот фрагмент следующим образом.

<pre>\$id = 4; \$sql = "SELECT `text` FROM `reviews` WHERE `id` = :id"; \$result = \$conn->prepare(\$sql); \$result->execute(['id' => \$id]); echo '<i>' . \$result->fetch(PDO::FETCH_OBJ)->text . '</i>';</pre>	<pre>12 \$id = 4; 13 \$sql = "SELECT `text` FROM `reviews` WHERE `id` = '{\$id}'"; 14 \$result = \$conn->query(\$sql); 15 16 echo '<i>' . \$result->fetch(PDO::FETCH_OBJ)->text . '</i>';</pre>
---	--

Включение значения с одинарными кавычками, фигурными скобками и знаком доллара заменяется внутри строки запроса на «имя», перед которым ставится двоеточие, т.е. на параметр:

```
WHERE `id` = '{$id}'  
WHERE `id` = :id
```

А выполнение одного метода query «раскладывается» на два действия:

```
было  
$result = $conn -> query($sql);  
СТАЛО  
$result = $conn -> prepare($sql);  
$result -> execute(['id' => $id]);
```

Т.е. мы должны сначала приготовить результат к использованию, а затем получить его, заполнив параметры конкретными значениями.

Приготовление-и-выполнение параметризованного запроса напоминает создание шаблона и его заполнение. Этот способ более устойчив к SQL-инъекциям и считается безопаснее. Можно приготовить один шаблон-запрос и далее многократно применять его, и эти применения будут быстрее повторяющихся запросов с разными данными.

Рассмотрим две инструкции:

```
$a = ['id' => $id];  
$a = ['id' => &$id];
```

Первая создаёт массив, в котором значением ключа id будет то значение, которое в момент выполнения этой инструкции находилось в переменной \$id.

Вторая создаёт массив, в котором ключ id будет синонимом переменной \$id (ссылкой) и при изменении значения в переменной id выражение \$a[id] будет возвращать изменённое значение.

```
$sql = "SELECT `text` FROM `reviews` WHERE `id` = :id";
$result = $conn -> prepare($sql);
$id = 3;
$a = ['id' => &$id];
$result -> execute($a);
echo '<p>' . $result -> fetch(PDO::FETCH_OBJ) -> text . '</p>';

$id = 4; $result -> execute($a);
echo '<p>' . $result -> fetch(PDO::FETCH_OBJ) -> text . '</p>';
$id = 5; $result -> execute($a);
echo '<p>' . $result -> fetch(PDO::FETCH_OBJ) -> text . '</p>';
```

← → ↻ ⓘ 92.53.66.41/mysql2prep2.php

Извлечение данных...

Мне понравился сайт, но мало данных

Несекурно, взломать пара пустяков

Тут светло!

Выше рассмотрены именованные параметры (с двоеточием). Чтобы их заполнить, на вход методу execute подаётся ассоциативный массив (вида ['key' => value]). Мы рассматривали извлечение значений из БД.

При вставке значений более удобны безымянные параметры. Они обозначаются просто знаками вопроса:

```
$r = ['Илья', 'elias@kodaktor.ru', 'Нужно совершенствоваться'];
$sql = "INSERT INTO `reviews` (`name`, `email`, `text`) VALUES (?, ?, ?)";

$result = $conn -> prepare($sql);
$result -> execute($r);
```

Тогда на вход методу execute подаётся обычный массив с числовыми индексами.

Выполним запрос, позволяющий выяснить самый последний id:

```
mysql> SELECT MAX( `id` ) FROM `reviews`;
```

```
+-----+
| MAX( `id` ) |
+-----+
|          4  |
+-----+
1 row in set (0.00 sec)
```

Вот как он выглядит в PHP-сценарии:

```
$q = 'MAX(`id`)';
$sql = "SELECT {$q} FROM `reviews`";
$result = $conn -> query($sql) -> fetch(PDO::FETCH_OBJ);
var_dump($result);
```

```
object(stdClass)#3 (1) {
    ["MAX(`id`)"] =>
        string(1) "8"
}
```

```
echo($result -> $q); // 8
```

Создадим сценарий, который извлекает запись с указанным id либо, если id не указан, то последнюю

```
<?php
error_reporting(E_ALL);
ini_set('display_errors', 1);
header('Content-type: text/html; charset=utf-8');
echo "<h1>Извлечение данных...</h1>\n";
$dbloc = '92.53.66.41';
$dbuser = $dbname = 'user1';
$dbpass = 'Qwerty123';

$dsn = "mysql:host={$dbloc};dbname={$dbname}";
$conn = new PDO($dsn, $dbuser, $dbpass);
$conn->exec('SET CHARACTER SET utf8');

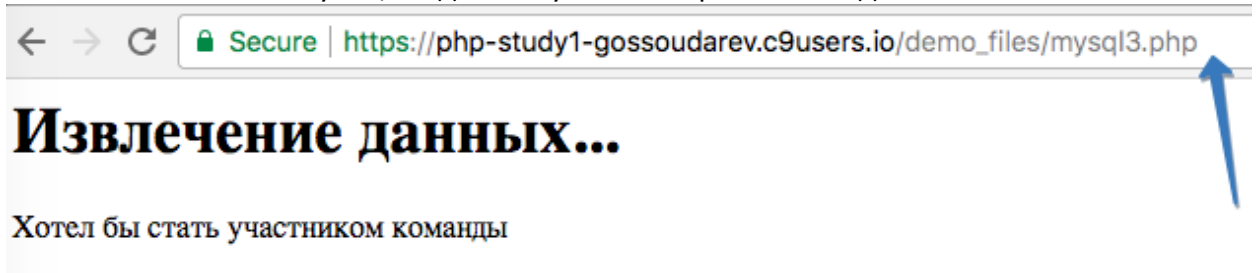
$q = 'MAX(`id`)';
$sql = "SELECT {$q} FROM `reviews`";
$result = $conn -> query($sql) -> fetch(PDO::FETCH_OBJ);
```



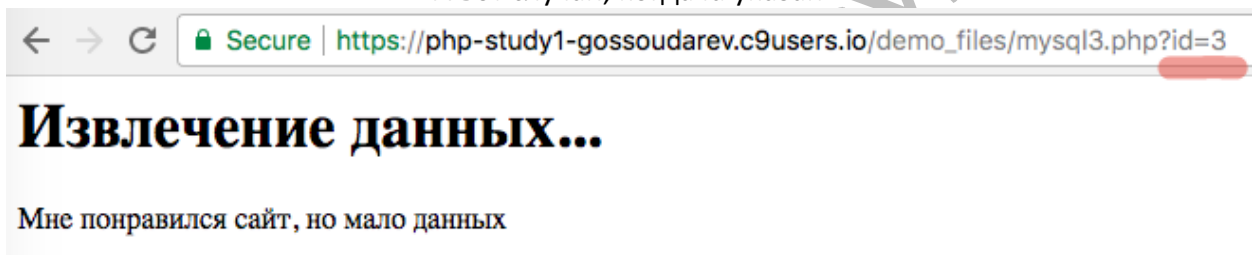
```
$id = $_GET['id'] ?? $result -> $q; // если не определен GET-параметр id, будет взята последняя запись
```

```
$sql = "SELECT `text` FROM `reviews` WHERE `id` = '{$id}'";  
$result = $conn -> query($sql) -> fetch(PDO::FETCH_OBJ);  
echo ($result -> text);
```

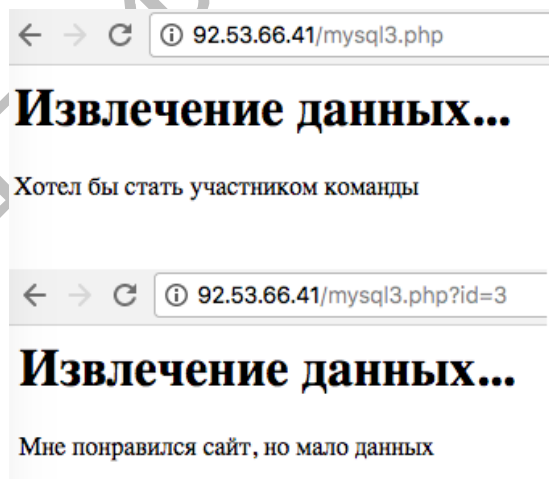
Вот случай, когда id не указан - берётся последняя запись



А вот случай, когда id указан



Сценарий, запущенный на c9.io



То же самое на облачном сервере

Код для добавления записи

```
<?php  
$dbloc = '92.53.66.41';  
$dbuser = $dbname = 'user1';  
$dbpass = 'Qwerty123';
```

```
$dsn = $dsn = "mysql:host={$dbloc};dbname={$dbname}";  
$conn = new PDO($dsn, $dbuser, $dbpass);  
$conn->exec('SET CHARACTER SET utf8');  
  
[$name, $email, $text] = ['Bill Gathes', 'pres@macrosoft.org', 'Готов проспонсировать ваш сайт'];  
$sql = "INSERT INTO `reviews` (`name`, `email`, `text`) VALUES ('{$name}', '{$email}', '{$text}')";  
$result = $conn -> query($sql);  
if ($result) echo ('OK');
```

Выделенные строки повторяются в каждом сценарии. Логично записать их в отдельный файл `bd.php` и вставлять во все нуждающиеся в них файлы с помощью `require_once`

bd.php

```
<?php  
$dbloc = '92.53.66.41';  
$dbuser = $dbname = 'user1';  
$dbpass = 'Qwerty123';  
$dsn = $dsn = "mysql:host={$dbloc};dbname={$dbname}";  
return new PDO($dsn, $dbuser, $dbpass);
```

get8.php

```
<?php  
echo '<h1>Извлечение данных...</h1>';  
$conn = require_once ('bd.php');  
$conn -> exec('SET CHARACTER SET utf8');  
  
$id = 8;  
$sql = "SELECT `text` FROM `reviews` WHERE `id` = '{$id}'";  
$result = $conn -> query($sql) -> fetch(PDO::FETCH_OBJ);  
echo ($result -> text);
```

Следующий этап - создать форму.

<https://kodaktor.ru/formreview1>

Имя

Email

Отзыв

☐ Согласен с условиями

Отправить отзыв

Стили к ней доступны по адресу [//kodaktor.ru/css/formstyle1](http://kodaktor.ru/css/formstyle1)

Мы можем отправлять данные приложению средствами самой формы или же реализовать AJAX-отправку. Первый вариант является исторически первым и как бы традиционным, с него и начнём.

Будем извлекать данные из формы примерно в таком стиле:

```
[$name, $email, $text] = [ $_GET['name'] ?? 'Unknown user', $_GET['email'] ?? 'anon@mail.ru',  
$_GET['text'] ?? 'Пустое сообщение'];
```

На самом деле мы не будем позволять отправлять пустую форму, но GET-запрос можно отправить и просто в адресной строке.

Мы сделаем один сценарий, принимающий данные из формы и добавляющий их в таблицу, и другой, выводящий список существующих отзывов.

Для вывода списка у нас есть тоже как минимум два варианта:

1. Выдавать готовую сформированную разметку (список, таблицу).
2. Выдавать JSON и предоставить клиентскому сценарию самостоятельно строить список или таблицу по нему (можно воспользоваться быстрой шаблонизацией типа Vue для распределения информации по HTML-элементам).

Воспользуемся первым вариантом.

Самый «лобовой» вариант решения:

```
echo '<table>';
while ($row = $result->fetch(PDO::FETCH_OBJ)) {
    echo '<tr>';
    echo '<td>' . $row->name . '</td>';
    echo '<td>' . $row->email . '</td>';
    echo '<td>' . $row->text . '</td>';
    echo '</tr>';
}
echo '</table>';
```

Но можно воспользоваться более элегантным подходом. Если мы могли бы получить данные в виде массива, то каждому элементу массива мы могли бы сопоставить (map) ячейку таблицы.

Рассмотрим вначале такую отдельную задачу: по массиву чисел сформировать список их квадратов (т.е. с элементами).

- 1
- 4
- 9
- 16
- 25

```
<ul>
<?php
    $a = [1, 2, 3, 4, 5];
    echo implode('', array_map(function($x){return '<li>'.$x**2 . '</li>';}, $a));
?>
</ul>
```

В JavaScript это выглядит так (https://kodaktor.ru/array_map):

```
let $a = [1, 2, 3, 4, 5];
document.body.innerHTML='<ul>'+$a.map(function($x){return '<li>'+$x**2+'</li>'}).join('')+ '</ul>';
или с использованием стрелок $a.map($x=>'<li>'+$x**2+'</li>').join('')
```

В PHP стиль здесь чисто функциональный (композиция применений), а в JavaScript объектно-функциональный (цепочка методов)...

Итак, наш шаблон - implode('', array_map(function(\$x){return '...'}))

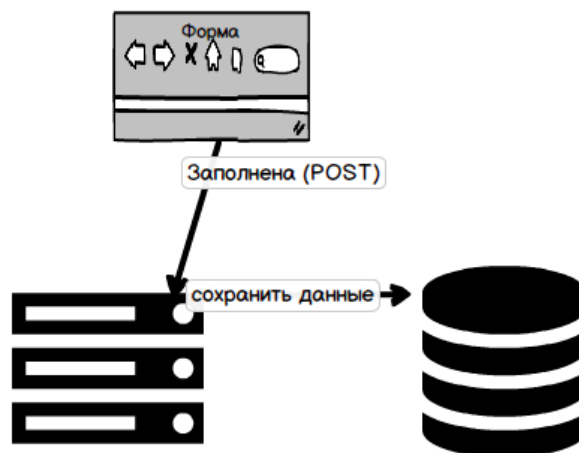
Аргумент \$x функции обратного вызова, которую array_map применяет к каждому элементу массива, пробегает эти элементы, а функция производит над ним некоторое действие, например, окружение тегами типа <td></td>.

Получается следующее:

```
echo '<table>';
while ($row = $result->fetch(PDO::FETCH_NUM)) {
    echo '<tr>'.implode(',',array_map(function($x){return '<td>'.$x.'</td>';},$row)).'</tr>';
}
echo '</table>';
```

Итак, у нас есть все необходимые части приложения, осталось только собрать их воедино.

На самом деле для такого небольшого приложения кажется наиболее удобным сосредоточить основную логику в одном файле (index.php), который будет делать разные вещи в зависимости от того, как к нему обращаются. Различие будет проходить по линии метода протокола HTTP. Если поступает POST-запрос, это значит, что была заполнена и отправлена форма. В этом **и только этом** случае нужно сохранить данные в таблицу reviews.



И в **любом** случае нам нужно отобразить форму и список уже существующих отзывов. Т.е. это происходит и в случае POST-запроса и в случае GET-запроса.

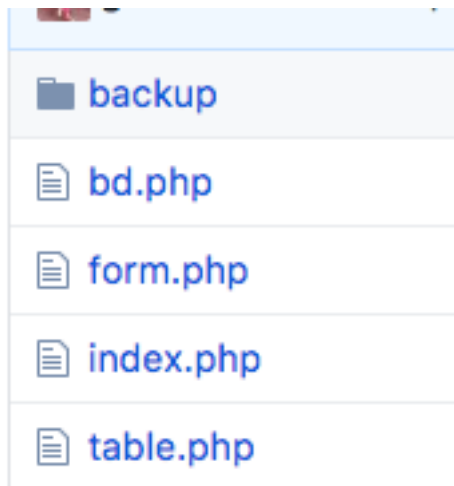
На таком примитивном уровне проектирования мы мыслим файлами как фрагментами кода, имеющими для удобства отдельное расположение и отдельные имена. Эти фрагменты включаются в основной файл с помощью `require_once` и все «живут» в одном пространстве имён. Это означает, что если где-то повыше в коде есть включение файла, в котором определяется переменная `$conn`, то ниже после включения она будет доступна и в основном файле и в любом другом, который будет включен ниже (речь не идёт о функциях!).

Итак, структура приложения при разбиении на файлы такова:

За отображение формы будет отвечать файл `form.php`, который содержит готовую разметку формы (а в принципе мог бы извлекать её из какого-то внешнего источника).

За отображения списка существующих отзывов будет отвечать файл `table.php`.

И, наконец, файлу `bd.php` мы поручим создание соединения с базой данных.



Замечание о безопасности

Предположим, посетитель вводит в форму нечто подобное этому:

Имя

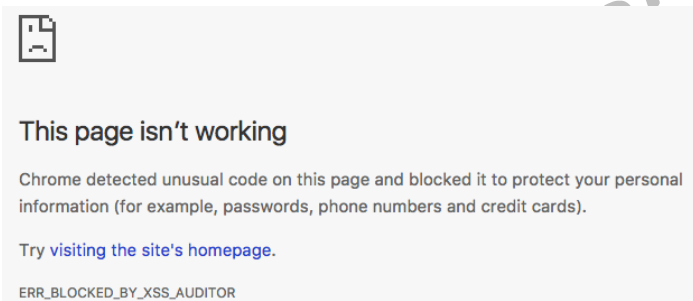
Email
Отзыв

<script>alert(1)</script>

☒ Согласен с условиями

Отправить отзыв

т.е. пытается осуществить XSS-атаку, внедряя в то, что по идее должно быть просто текстом, то, что является сценарием на JavaScript. Если не предпринимать специальных мер, это будет прямо так записано в базу данных, а при выводе... мы увидели бы в данном случае всплывающее окно с единицей... или такое предупреждение:



Суперглобальные переменные часто используются злоумышленниками, пытающимися отыскать средства для атаки и вмешательства в работу сайта. Они загружают в `$_POST`, `$_GET` или в другие суперглобальные переменные вредоносный код, в том числе команды UNIX или MySQL, которые, если по незнанию к ним обратиться, могут разрушить или отобразить незащищенные данные.

Именно поэтому перед применением суперглобальных переменных их всегда следует подвергать предварительной обработке. Для этого можно воспользоваться PHP-функцией `htmlspecialchars`. Она занимается преобразованием всех символов в элементы HTML.

Например, символы «меньше» и «больше» (`<` и `>`) превращаются в строки `<` и `>`, то же самое делается для перевода в безопасное состояние всех кавычек, обратных слешей и т. д. В результате вышеприведённая «атака» запишется в базе данных так:

| 10 | Ilya Gosudarev | ilia.gossoudarev@gmail.com | <script>alert(1)</script>

```
$r = array_map(function($x){return htmlentities($x);}, [ $_POST['name'] ?? 'Unknown user',  
$_POST['email'] ?? 'anon@mail.ru', $_POST['text'] ?? 'Пустое сообщение']);
```

Т.е. мы здесь собираем массив из значений, пришедших методом POST и прогоняем его через функцию htmlentities.

Работающее приложение:
<http://92.53.66.41/reviews/>

На уровне взаимодействия с базой данных от инъекций вредоносного кода нас защищают подготовленные запросы.

index.php

```
1 <meta charset="utf-8"><link rel="stylesheet" href="//kodaktor.ru/css/formstyle1">  
2 <h1>Работа с отзывами... </h1>  
3 <?php  
4     $conn = require_once ('bd.php');  
5     $conn -> exec('SET CHARACTER SET utf8');  
6     $f = '<h2>Форма для заполнения... </h2>';  
7     if ($_SERVER['REQUEST_METHOD']==='POST') {  
8         $r = array_map(function($x){ return htmlentities($x);},  
9             [  
10                 $_POST['name'] ?? 'Unknown user',  
11                 $_POST['email'] ?? 'anon@mail.ru',  
12                 $_POST['text'] ?? 'Пустое сообщение'  
13             ]  
14         );  
15         $sql = "INSERT INTO `reviews` (`name`, `email`, `text`) VALUES (?, ?, ?);";  
16         $result = $conn -> prepare($sql);  
17         $result -> execute($r);  
18         if ($result) {  
19             $f = '<style>.right {width: 60%; margin-left: 35%; zoom: 80%;}</style>';  
20             $f .= '<div class="right"><h2>Добавить ещё один отзыв...</h2></div>';  
21             $i = '<h3>Данные успешно добавлены, спасибо!</h3>';  
22             $log = fopen('log.txt', 'a'); fwrite($log, $conn -> lastInsertId());  
23             fwrite($log, "\n"); fclose($log);  
24         } else {  
25             $i = '<h4>Что-то не так!</h4>';  
26         }  
27         echo $i;  
28     }  
29     echo $f;  
30     echo "<div class=\"right\">";  
31     echo "<form method=\"post\" action=\"\"/{$_SERVER['SERVER_NAME']}{$_SERVER['SCRIPT_NAME']}\>";  
32     echo require_once ('form.php');  
33     echo "</form></div>";  
34     echo "<h2>Список отзывов... </h2>\n";  
35     echo require_once ('table.php');
```


bd.php

```
1 <?php
2 $dbloc = '92.53.66.41' ;
3 $dbuser = $dbname = 'user1';
4 $dbpass = 'Qwerty123';
5 $dsn = $dsn = "mysql:host={$dbloc};dbname={$dbname}";
6 return new PDO($dsn, $dbuser, $dbpass);
```

table.php

```
table.php x
1 <?php
2 $sql = "SELECT `name`,`email`,`text` FROM `reviews` ORDER BY `id` DESC;";
3 $result = $conn -> query($sql);
4 $list = '';
5 $list .= '<table>';
6 while ($row = $result->fetch(PDO::FETCH_NUM)) {
7     $list .= '<tr>'.implode(' ',array_map(function($x){return '<td>'.$x.'</td>';},$row)).'</tr>';
8 }
9 $list .= '</table>';
10 return $list;
```

и form.php

```
form.php x
1 <?php
2 return <<<_END
3     <div class="field">
4         <label class="label">Имя</label>
5         <div class="control">
6             <input class="input" id="name" name="name" required="required" type="text">
7         </div>
8     </div>
9     <div class="field">
10        <label class="label">Email</label>
11        <div class="control">
12            <input class="input" id="email" name="email" required="required" type="email">
13        </div>
14    </div>
15    <div class="field">
16        <label class="label">Отзыв</label>
17        <div class="control">
18            <textarea class="textarea" id="text" name="text" required="required"></textarea>
19        </div>
20    </div>
21    <div class="field">
22        <div class="control">
23            <label class="checkbox"><input required="required" type="checkbox"> Согласен с условиями</label>
24        </div>
25    </div>
26    <div class="field">
27        <div class="control has-text-centered">
28            <button class="button is-warning has-text-weight-bold" type="submit">Отправить отзыв</button>
29        </div>
30    </div>
31 _END;
```

(с)2017 Илья Государев

Работа с отзывами...

Данные успешно вставлены!

Добавить ещё один отзыв...

Имя

Email

Отзыв

☐ Согласен с условиями

Отправить отзыв

Список отзывов...

Фёдор	phoedaur@gmail.com	Полностью поддерживаю предыдущие отзывы. Это моё мнение тоже...
Ilya Gosudarev	ilia.gossoudarev@gmail.com	Хочу работать в вашей команде!
little me	lopez@valdo.ch	Надо улучшать интерфейс
light	light@dark.net	Тут светло!
Анонимус	guy.fawkes@protonmail.com	Стало лучше
Sasha	sasha@mail.ru	Мне понравился сайт, но мало данных
Petr Ivanov	user@yandex.ru	Шикарно!
elias	elias.b.goss@gmail.com	Очень хороший сайт!

Отметим, что в этом приложении используются самые базовые концепции, и достаточно примитивным образом. Запросы в БД отправляются без участия модели напрямую SQL-запросами, нет шаблонизации и явного использования каких-либо шаблонов проектирования.

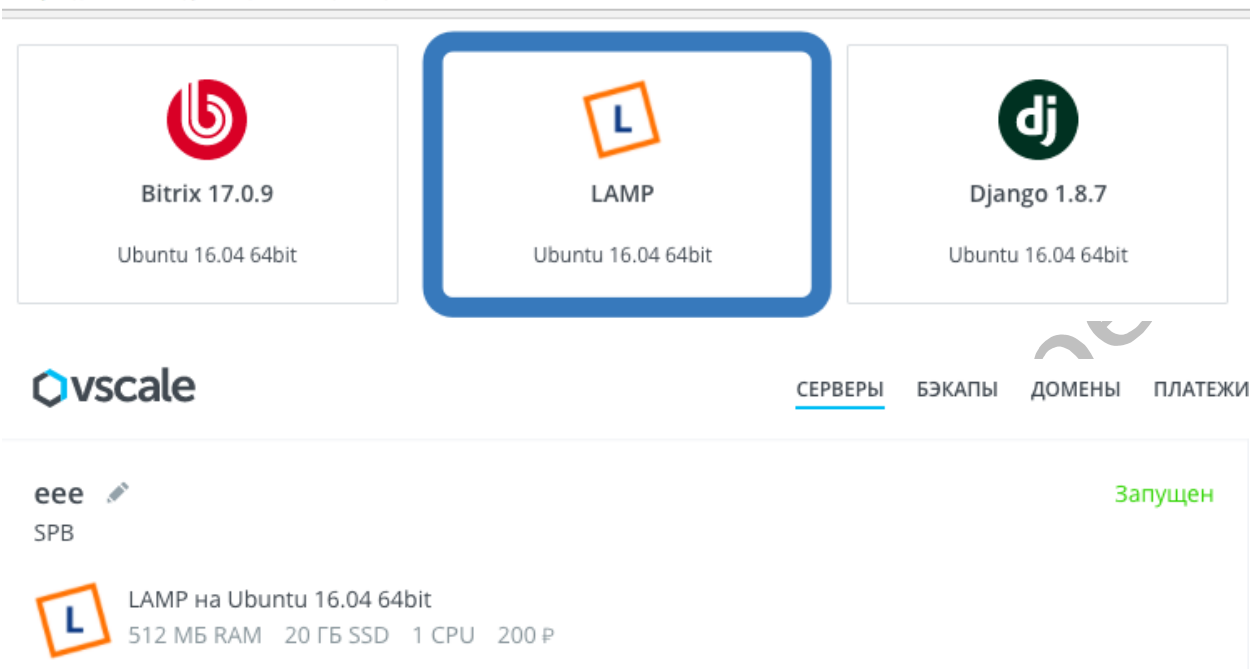
В целом же задачи создания таких систем как блоги решаются на сегодняшний день наиболее эффективно двумя способами:

- а) вручную с помощью фреймворков типа Laravel.
- б) развёртыванием готового приложения на основе системы управления контентом типа Wordpress.

Дополнение 1

Создадим виртуальный сервер на Ubuntu16 и с приложением LAMP:

<https://vscale.io/panel/scalelets/new/>



После создания сервера на почту приходит информация о логине и пароле на SSH и на приложение (т.е. в данном случае MySQL).

Подключаемся к серверу, загрузив в него заранее сгенерированные ключи.

ssh root@92.53.66.41

На сайте github.com создаём репозиторий php_reviews Далее будем отправлять код в репозиторий с локального компьютера а на удалённом сервере его получать и запускать.	
На компьютере разработчика	На vscale.io
git remote add origin https://github.com/myRepo/php_reviews.git git add -A git status git commit -m 'first' git push -u origin master git checkout -b step1 git add .	apt-get install git git clone https://github.com/myRepo/php_reviews.git . git checkout -b prepare

git push -u origin prepare ...	git pull origin prepare
-----------------------------------	-------------------------

Подключаемся в первый раз на удалённом сервере к запущенному там MySQL:
root@cs186959:/var/www/html# **mysql -u'root' -p'tmro1zkfmb'**

Убеждаемся, что от имени root видим соответствующие базы данных:

mysql> **show databases;**

```
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
```

4 rows in set (0.01 sec)

Разрешаем внешний доступ к БД

```
nano /etc/mysql/mysql.conf.d/mysqld.cnf
закомментировать строку с bind #bind-address = 127.0.0.1
service mysql restart
```

для каждого пользователя создаём базу данных - одноимённую.

```
CREATE DATABASE `user1` DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;
use mysql;
CREATE USER 'user1'@'%' IDENTIFIED BY 'Qwerty123';
GRANT ALL PRIVILEGES ON user1.* TO 'user1'@'%';
use user1;
CREATE TABLE `reviews` ( `id` int(11) NOT NULL AUTO_INCREMENT, `name` text COLLATE
utf8_unicode_ci NOT NULL, `email` text COLLATE utf8_unicode_ci NOT NULL, `text` text COLLATE
utf8_unicode_ci NOT NULL, PRIMARY KEY (`id`)) ENGINE=InnoDB AUTO_INCREMENT=7
DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
INSERT INTO `reviews` VALUES (1,'elias','elias.b.goss@gmail.com','Очень хороший
сайт!'),(2,'Petr Ivanov','user@yandex.ru','Шикарно!'),(3,'Sasha','sasha@mail.ru','Мне
понравился сайт, но мало данных'),(4,'Анонимус','guy.fawkes@protonmail.com','Несекурно,
взломать пара пустяков'),(5,'light','light@dark.net','Тут светло!'),(6,'little
me','lopez@valdo.ch','Надо улучшать интерфейс');
SELECT * FROM `reviews` WHERE 1;
UPDATE `reviews` SET `text`='Стало гораздо лучше' WHERE `id`=4;
```

Всегда можно быстро убрать и таблицу и базу:

```
DROP TABLE `reviews`;
DROP DATABASE gossouda_guest;
```

Далее можно протестировать вход

```
mysql -u'user1' -p'Qwerty123'
```

или извне

```
mysql -h'92.53.66.41' -u'user1' -p'Qwerty123'
```

и далее поработать с БД

```
mysql> use user1;
```

```
mysql> UPDATE `reviews` SET `text`='Ведётся большая работа над сайтом' WHERE `id`='7';
```

```
mysqldump -u'user1' -p'Qwerty123' user1 > backup.sql
```

после создания пользователя сервис mysql нужно перезапустить

(с)2017 Илья Государев

Дополнение 2

Предположим, мы хотим, чтобы id был непрерывен, т.е. не возникало дыр вида 1, 2, 3, 5

Если исходить из предположения о том, что до сих пор последовательность была непрерывной, и мы удаляем последнюю запись, то $\max(id)$ всегда равен общему количеству записей, поэтому

1. пусть $n = \text{select max(id) from reviews};$
2. $\text{delete from reviews where id}=n;$
3. $\text{ALTER TABLE reviews AUTO_INCREMENT} = n;$
4. вставка новой

При (перед) вставке новой записи AUTO_INCREMENT должен быть сделан на 1 больше, чем

Допустим были записи 1, 2, 3, 4

1. $n = \text{select max(id)}$ и следовательно $n = 4$
сейчас AUTO_INCREMENT равен $n+1=5$ для следующей записи, которая будет вставлена
2. удаляем запись $n=4$ т.е. последнюю
3. после этого $\max(id)$ равен $n-1 = 3$, AUTO_INCREMENT же **остаётся** равен $n+1=5$
но мы поправляем: делаем его равным $(n-1)+1 = n$, т.е. 4
4. и при вставке новой записи она пойдёт под номером 4.

эта операция полностью эквивалентна update при которой меняется всё кроме id.

посмотреть который будет назначен следующей вставляемой записи можно так:

```
SELECT `AUTO_INCREMENT` FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_SCHEMA =  
DATABASE() AND TABLE_NAME='reviews';
```

В PHP после вставки это будет выглядеть так: `$conn -> lastInsertId();` но это последний id, а предыдущая команда выдаёт уже увеличенный на единицу, так сказать, готовый к применению.