Практика по работе с БД в PHP

Пусть в базе данных есть такая таблица workers:

id	name	age	salary
1	Коля	23	400
2	Вася	24	500
3	Петя	25	600

Давайте выведем ее в таком виде в браузер.

Для этого средствами PHP нам надо сформировать следующий HTML код:

```
<table>
    <tr>
        <th>id</th>
        <th>name</th>
        <th>age</th>
        <th>salary</th>
    </tr>
    <tr>
        <td>1</td>
        <td>Коля</td>
        <td>23</td>
        <td>400</td>
    </tr>
    <tr>
        <td>2</td>
        <td>Вася</td>
        <td>24</td>
        <td>500</td>
```

```
    </tr>
    <tr>
        <td>3</td>
        <td>Петя</td>
        <td>25</td>
        <td>600</td>
    </tr>
</table>
```

Каким образом мы это сделаем: часть HTML кода мы наберем вручную, а часть за нас сформирует PHP.
Вручную мы наберем статическую часть HTML - тег table и первую tr с заголовками таблицы.
А вот собственно ряды таблицы пусть сформирует PHP, взяв данные из БД.
Наберем статическую часть HTML кода и подготовим в нем место для вставки PHP:

```
<table>
    <tr>
        <th>id</th>
        <th>name</th>
        <th>age</th>
        <th>salary</th>
    </tr>
    <?php
        //тут будет PHP код, который сформирует эту часть таблицы
    ?>
</table>
```

Наш PHP код должен отправить запрос к базе данных, достать массив работников, затем сформировать из него соответствующее количество tr с td-шками.
Давайте достанем все работников из таблицы workers и запишем их в массив **$data** (пусть подключение к БД выполнено где-то выше, не будем его записывать для краткости):

```
<table>
    <tr>
        <th>id</th>
        <th>name</th>
        <th>age</th>
        <th>salary</th>
    </tr>
    <?php
        $query = "SELECT * FROM workers";
        $result = mysqli_query($link, $query) or die( mysqli_error($link) );
        for ($data = []; $row = mysqli_fetch_assoc($result); $data[] = $row);
        var_dump($data);
    ?>
</table>
```

Полученный нами массив будет выглядеть так:

```
[
    ['id' => '1', 'name' => 'Коля', 'age' => '23', 'salary' => '400'],
    ['id' => '2', 'name' => 'Вася', 'age' => '24', 'salary' => '500'],
    ['id' => '3', 'name' => 'Петя', 'age' => '25', 'salary' => '600'],
]
```

Давайте переберем его циклом foreach, сформировав при этом tr-ки и td-шки:

```
<table>
    <tr>
        <th>id</th>
```

```
            <th>name</th>
            <th>age</th>
            <th>salary</th>
        </tr>
        <?php
            $query = "SELECT * FROM workers";
            $result = mysqli_query($link, $query) or die(mysqli_error($link));
            for ($data = []; $row = mysqli_fetch_assoc($result); $data[] = $row);

            $result = '';
            foreach ($data as $elem) {
                $result .= '<tr>';

                $result .= '<td>' . $elem['id'] . '</td>';
                $result .= '<td>' . $elem['name'] . '</td>';
                $result .= '<td>' . $elem['age'] . '</td>';
                $result .= '<td>' . $elem['salary'] . '</td>';

                $result .= '</tr>';
            }

            echo $result;
        ?>
    </table>
```

Наша задача решена. Если запустить этот код (не забыв про подключение к БД), то на экран выведется соответствующая таблица.

При этом, конечно же, наша HTML таблица будет формироваться из тех данных, которые есть в текущий момент в БД. То есть если там будет больше записей - то и в браузере будет выведено больше записей.
В предыдущем уроке мы вывели список работников в виде таблицы HTML.
Давайте модифицируем эту задачу, добавив возможность удаления работников.
Для этого добавим в таблицу еще одну колонку, в которой для каждого работника будет размещаться ссылка на удаление работника:

```
<table>
    <tr>
        <th>id</th>
        <th>name</th>
        <th>age</th>
        <th>salary</th>
        <th>delete</th>
    </tr>
    <tr>
        <td>1</td>
        <td>Коля</td>
        <td>23</td>
        <td>400</td>
        <td><a href="">удалить</a></td>
    </tr>
    <tr>
        <td>2</td>
        <td>Вася</td>
        <td>24</td>
        <td>500</td>
```

```
            <td><a href="">удалить</a></td>
        </tr>
        <tr>
            <td>3</td>
            <td>Петя</td>
            <td>25</td>
            <td>600</td>
            <td><a href="">удалить</a></td>
        </tr>
    </table>
```

Сделаем так, чтобы при переходе по ссылке мы попадали на ту же страницу браузера, но отправляя при этом GET запрос с id работника, которого мы хотим удалить:

```
    <table>
        <tr>
            <th>id</th>
            <th>name</th>
            <th>age</th>
            <th>salary</th>
            <th>delete</th>
        </tr>
        <tr>
            <td>1</td>
            <td>Коля</td>
            <td>23</td>
```

```
            <td>400</td>
            <td><a href="?del=1">удалить</a></td>
        </tr>
        <tr>
            <td>2</td>
            <td>Вася</td>
            <td>24</td>
            <td>500</td>
            <td><a href="?del=2">удалить</a></td>
        </tr>
        <tr>
            <td>3</td>
            <td>Петя</td>
            <td>25</td>
            <td>600</td>
            <td><a href="?del=3">удалить</a></td>
        </tr>
    </table>
```

Как это будет работать: если мы перейдем, например, по ссылке для работника "Коля", то отправим GET запросом параметр del со значением 1, соответствующим id Коли в таблице базы данных.

В коде PHP мы можем получить id работника для удаления, обратившись к $_GET['del'] и затем удалить его, вот так:

```
<?php
    $del = $_GET['del']; // получим id для удаления
    $query = "DELETE FROM workers WHERE id=$del"; // сформируем запрос на удаление
```

```
        mysqli_query($link, $query) or die(mysqli_error($link)); // удалим
?>
```

Так как мы не всегда выполняем операцию удаления, то GET параметра может и не быть в адресной строке. Поэтому давайте проверим его наличие с помощью функции isset - и только, если параметр есть - будем выполнять удаление:

```
<?php
        if (isset($_GET['del'])) {
                $del = $_GET['del'];
                $query = "DELETE FROM workers WHERE id=$del";
                mysqli_query($link, $query) or die(mysqli_error($link));
        }
?>
```

Давайте теперь вспомним код для вывода данных в виде HTML таблицы, полученный нами в предыдущем уроке:

```
<table>
        <tr>
                <th>id</th>
                <th>name</th>
                <th>age</th>
                <th>salary</th>
        </tr>
        <?php
                $query = "SELECT * FROM workers";
                $result = mysqli_query($link, $query) or die( mysqli_error($link) );
                for ($data = []; $row = mysqli_fetch_assoc($result); $data[] = $row);
```

```
            $result = '';
            foreach ($data as $elem) {
                $result .= '<tr>';
                $result .= '<td>' . $elem['id'] . '</td>';
                $result .= '<td>' . $elem['name'] . '</td>';
                $result .= '<td>' . $elem['age'] . '</td>';
                $result .= '<td>' . $elem['salary'] . '</td>';
                $result .= '<td>' . $elem['salary'] . '</td>';
                $result .= '</tr>';
            }
            echo $result;
        ?>
    </table>
```

Добавим в таблицу еще одну ячейку со ссылкой на удаление (пока без GET запроса):

```
    <table>
        <tr>
            <th>id</th>
            <th>name</th>
            <th>age</th>
            <th>salary</th>
            <th>delete</th>
        </tr>
```

```
    <?php
        $query = "SELECT * FROM workers";
        $result = mysqli_query($link, $query) or die( mysqli_error($link) );
        for ($data = []; $row = mysqli_fetch_assoc($result); $data[] = $row);

        $result = '';
        foreach ($data as $elem) {
            $result .= '<tr>';

            $result .= '<td>' . $elem['id'] . '</td>';
            $result .= '<td>' . $elem['name'] . '</td>';
            $result .= '<td>' . $elem['age'] . '</td>';
            $result .= '<td>' . $elem['salary'] . '</td>';
            $result .= '<td>' . $elem['salary'] . '</td>';
            $result .= '<td><a href="">удалить</a></td>';

            $result .= '</tr>';
        }

        echo $result;
    ?>
</table>
```

Давайте теперь сделаем так, чтобы при переходе по ссылке передавался GET запрос на удаление. Так как наши tr-ки формируются в цикле, мы не можем просто вручную проставить номера id для удаления в GET запрос. Пусть это сделает PHP автоматически.
Сам id работника хранится в $elem['id'] - подставим это значение в GET запрос в href ссылки, вот так:

```
<table>
```

```
        <tr>
            <th>id</th>
            <th>name</th>
            <th>age</th>
            <th>salary</th>
            <th>delete</th>
        </tr>
        <?php
            $query = "SELECT * FROM workers";
            $result = mysqli_query($link, $query) or die( mysqli_error($link) );
            for ($data = []; $row = mysqli_fetch_assoc($result); $data[] = $row);

            $result = '';
            foreach ($data as $elem) {
                $result .= '<tr>';

                $result .= '<td>' . $elem['id'] . '</td>';
                $result .= '<td>' . $elem['name'] . '</td>';
                $result .= '<td>' . $elem['age'] . '</td>';
                $result .= '<td>' . $elem['salary'] . '</td>';
                $result .= '<td>' . $elem['salary'] . '</td>';
                $result .= '<td><a href="?del=' . $elem['id'] . '">удалить</a></td>';

                $result .= '</tr>';
```

```
        }

        echo $result;
    ?>
</table>
```

Теперь при переходе по ссылке будет происходить передача GET параметра с id работника, которого мы хотим удалить. Но самого удаления пока не будет, так как мы не выполняем SQL запрос на удаление.

Давайте добавим код для удаления, полученный нами в начале урока. Учтите, что его нужно добавлять **до получения** работников из БД, чтобы при удалении работник сначала удалился из базы, а уже потом мы получили оставшихся и вывели их на экран:

```
<table>
    <tr>
        <th>id</th>
        <th>name</th>
        <th>age</th>
        <th>salary</th>
        <th>delete</th>
    </tr>
    <?php
        // Удаление по id (до получения!):
        if (isset($_GET['del'])) {
            $del = $_GET['del'];
            $query = "DELETE FROM workers WHERE id=$del";
            mysqli_query($link, $query) or die(mysqli_error($link));
        }

```

```
        // Получение всех работников:
        $query = "SELECT * FROM workers";
        $result = mysqli_query($link, $query) or die(mysqli_error($link));
        for ($data = []; $row = mysqli_fetch_assoc($result); $data[] = $row);

        // Вывод на экран:
        $result = '';
        foreach ($data as $elem) {
            $result .= '<tr>';
            $result .= '<td>' . $elem['id'] . '</td>';
            $result .= '<td>' . $elem['name'] . '</td>';
            $result .= '<td>' . $elem['age'] . '</td>';
            $result .= '<td>' . $elem['salary'] . '</td>';
            $result .= '<td>' . $elem['salary'] . '</td>';
            $result .= '<td><a href="?del=' . $elem['id'] . '">удалить</a></td>';
            $result .= '</tr>';
        }
        echo $result;
    ?>
</table>
```

Если запустить этот код, то в браузере мы увидим список работников. Если затем перейти по ссылке для удаления какого-либо работника, страница браузера перезагрузится (т.к. мы перешли по ссылке), работник удалится из базы и в таблице его уже не будет.

Еще раз: удаление следует размещать до получения работников из БД, иначе вы сначала получите всех работников вместе с тем, которого хотели удалить, только затем удалите его в базе, но в

HTML таблице работник не удалится. Его удаление произойдет только после перезагрузки страницы. Учтите это и не совершайте такой ошибки.
Давайте теперь сделаем добавление нового работника с помощью формы.
Давайте сделаем HTML код формы добавления:

```
<form action="" method="POST">
    <input name="name">
    <input name="age">
    <input name="salary">
    <input type="submit" value="добавить работника">
</form>
```

А теперь напишем PHP код, который будет формировать INSERT запрос для сохранения данных из формы в базу данных:

```
<?php
    $name = $_POST['name'];
    $age = $_POST['age'];
    $salary = $_POST['salary'];

    $query = "INSERT INTO workers SET name='$name', age='$age', salary='$salary'";
    mysqli_query($link, $query) or die(mysqli_error($link));
?>
```

Давайте теперь совместим нашу форму и код, обрабатывающий данные с нее в одном файле.
Напомню, что в этом случае наш скрипт выполнится два раза: первый раз пользователь зайдет на страницу, заполнит форму и отправит ее. После отправления мы попадем на эту же страницу и скрипт начнет выполнятся сначала, но уже будут доступны данные из формы.
Эти данные будут хранится в $_POST, если форма, как у нас, отправлена методом POST.
Наш PHP код для сохранения должен выполнится после того, как форма была отправлена и не должен выполнятся при первом заходе пользователя на страницу.
Добиться этого мы можем с помощью ифа. Например, можно спросить, не пустой ли POST и, если не пустой - только тогда начинать выполнять наш код для сохранения, вот так:

```
<?php
    if (!empty($_POST)) {
        $name = $_POST['name'];
```

```
        $age = $_POST['age'];
        $salary = $_POST['salary'];

        $query = "INSERT INTO workers SET name='$name', age='$age', salary='$salary'";
        mysqli_query($link, $query) or die(mysqli_error($link));
    }
?>
```

Давайте совместим наш PHP и форму:

```
<?php
    if (!empty($_POST)) {
        $name = $_POST['name'];
        $age = $_POST['age'];
        $salary = $_POST['salary'];

        $query = "INSERT INTO workers SET name='$name', age='$age', salary='$salary'";
        mysqli_query($link, $query) or die(mysqli_error($link));
    }
?>

<form action="" method="POST">
    <input name="name">
    <input name="age">
    <input name="salary">
    <input type="submit" value="добавить работника">
</form>
```

Я поставил форму под PHP кодом, но ее можно разместить где угодно - ведь наш файл со скриптом выполняется два раза и поэтому разницы нет, где что будет размещено.

В общем код, решающий поставленную задачу, написан. Его можно разместить на отдельной странице, а можно совместить с нашей HTML таблицей работников.
Первый вариант у нас уже реализован - можете потестировать его, разместив этот код в отдельном файле и проверив его работу (не забудьте про подключение к базе данных, которое я опускаю для краткости).
Давайте теперь совместим вывод таблицы работников с добавлением нового.
Вспомним код, который делает вывод таблицы и удаление:

```
<table>
    <tr>
        <th>id</th>
        <th>name</th>
        <th>age</th>
        <th>salary</th>
        <th>delete</th>
    </tr>
    <?php
        // Удаление по id (до получения!):
        if (isset($_GET['del'])) {
            $del = $_GET['del'];
            $query = "DELETE FROM workers WHERE id=$del";
            mysqli_query($link, $query) or die(mysqli_error($link));
        }

        // Получение всех работников:
        $query = "SELECT * FROM workers";
        $result = mysqli_query($link, $query) or die(mysqli_error($link));
        for ($data = []; $row = mysqli_fetch_assoc($result); $data[] = $row);
```

```
        // Вывод на экран:
        $result = '';
        foreach ($data as $elem) {
            $result .= '<tr>';
            $result .= '<td>' . $elem['id'] . '</td>';
            $result .= '<td>' . $elem['name'] . '</td>';
            $result .= '<td>' . $elem['age'] . '</td>';
            $result .= '<td>' . $elem['salary'] . '</td>';
            $result .= '<td>' . $elem['salary'] . '</td>';
            $result .= '<td><a href="?del=' . $elem['id'] . '">удалить</a></td>';
            $result .= '</tr>';
        }
        echo $result;
    ?>
</table>
```

Давайте совместим его с нашим кодом, который делает добавление нового работника. При этом форму добавления поставим под таблицу (ее можно поставить куда угодно), а сам PHP код добавления разместим до получения всех работников, чтобы сначала добавлялся новый и потом получались все работники вместе с новым добавленным.

Итак, вот наш код:

```
<table>
    <tr>
        <th>id</th>
        <th>name</th>
```

```
            <th>age</th>
            <th>salary</th>
            <th>delete</th>
        </tr>
        <?php
            // Сохранение нового (до получения!):
            if (!empty($_POST)) {
                $name = $_POST['name'];
                $age = $_POST['age'];
                $salary = $_POST['salary'];

                $query = "INSERT INTO workers SET name='$name', age='$age',
salary='$salary'";
                mysqli_query($link, $query) or die(mysqli_error($link));
            }

            // Удаление по id (до получения!):
            if (isset($_GET['del'])) {
                $del = $_GET['del'];
                $query = "DELETE FROM workers WHERE id=$del";
                mysqli_query($link, $query) or die(mysqli_error($link));
            }

            // Получение всех работников:
```

```
        $query = "SELECT * FROM workers";
        $result = mysqli_query($link, $query) or die(mysqli_error($link));
        for ($data = []; $row = mysqli_fetch_assoc($result); $data[] = $row);

        // Вывод на экран:
        $result = '';
        foreach ($data as $elem) {
            $result .= '<tr>';
            $result .= '<td>' . $elem['id'] . '</td>';
            $result .= '<td>' . $elem['name'] . '</td>';
            $result .= '<td>' . $elem['age'] . '</td>';
            $result .= '<td>' . $elem['salary'] . '</td>';
            $result .= '<td>' . $elem['salary'] . '</td>';
            $result .= '<td><a href="?del=' . $elem['id'] . '">удалить</a></td>';
            $result .= '</tr>';
        }
        echo $result;
    ?>
</table>
<form action="" method="POST">
    <input name="name">
    <input name="age">
```

```
        <input name="salary">
        <input type="submit" value="добавить работника">
</form>
```

Сейчас мы с вами начнем добавлять новый функционал к нашему скрипту и для простоты изложения давайте вынесем добавление нового работника на отдельную страницу, назовем ее add.php.
Вот ее код:

```
<?php
    if (!empty($_POST)) {
        $name = $_POST['name'];
        $age = $_POST['age'];
        $salary = $_POST['salary'];

        $query = "INSERT INTO workers SET name='$name', age='$age', salary='$salary'";
        mysqli_query($link, $query) or die(mysqli_error($link));
    }
?>
<form action="" method="POST">
    <input name="name">
    <input name="age">
    <input name="salary">
    <input type="submit" value="добавить работника">
</form>
```

Сейчас наш скрипт добавления работает так: пользователь нашего сайта вводит данные в форму, нажимает на кнопку отправки, страница сайта перезагружается и при этом выполняется INSERT запрос на добавление работника.
Если вы запустите этот код, то увидите, что инпуты после перезагрузки страницы очищаются. Давайте сделаем так, чтобы данные в инпутах оставались после отправки формы.

Используем для этого атрибут value, в который будем выводить соответствующие данные из массива $_POST.
Под соответствующими данными подразумевается то, что для инпута с именем name, в атрибут value мы вставим данные из $_POST['name'], вот так:

```
<input name="name" value="<?php echo $_POST['name']; ?>">
```

Для каждого инпута получается аналогично содержимое $_POST с именем соответствующего инпута:

```
<form action="" method="POST">
    <input name="name" value="<?php echo $_POST['name']; ?>">
    <input name="age" value="<?php echo $_POST['age']; ?>>
    <input name="salary" value="<?php echo $_POST['salary']; ?>>
    <input type="submit" value="добавить работника">
</form>
```

В нашем коде, однако, есть проблема - при первом заходе на страницу, когда форма не отправлена - мы увидим ошибки, так как в $_POST['name'], в $_POST['age'] и в $_POST['salary'] ничего не лежит, ведь массив $_POST - пустой.
Чтобы избежать подобной проблемы, давайте добавим иф с проверкой:

```
<form action="" method="POST">
    <input name="name" value="<?php if (isset($_POST['name'])) echo $_POST['name']; ?>">
    <input name="age" value="<?php if (isset($_POST['age'])) echo $_POST['age']; ?>>
    <input name="salary" value="<?php if (isset($_POST['salary'])) echo $_POST['salary']; ?>>
    <input type="submit" value="добавить работника">
</form>
```

Давайте добавим в наш код PHP часть для сохранения нового работника в БД:

```
<?php
    if (!empty($_POST)) {
        $name = $_POST['name'];
        $age = $_POST['age'];
        $salary = $_POST['salary'];
```

```
        $query = "INSERT INTO workers SET name='$name', age='$age', salary='$salary'";
        mysqli_query($link, $query) or die(mysqli_error($link));
    }
?>
<form action="" method="POST">
    <input name="name" value="<?php if (isset($_POST['name'])) echo $_POST['name']; ?>">
    <input name="age" value="<?php if (isset($_POST['age'])) echo $_POST['age']; ?>>
    <input name="salary" value="<?php if (isset($_POST['salary'])) echo $_POST['salary']; ?>>
    <input type="submit" value="добавить работника">
</form>
```

Если вы запустите этот код, введете данные в инпуты и нажмете на кнопку отправки - форма отправится, страница обновится, но данные из инпутов не пропадут.
Сейчас наша форма выглядит так:

```
<form action="" method="POST">
    <input name="name" value="<?php if (isset($_POST['name'])) echo $_POST['name']; ?>">
    <input name="age" value="<?php if (isset($_POST['age'])) echo $_POST['age']; ?>">
    <input name="salary" value="<?php if (isset($_POST['salary'])) echo $_POST['salary'];
?>">
    <input type="submit" value="добавить работника">
</form>
```

Ее код достаточно неудобен из-за того, что для добавления нового инпута, нам приходится писать его имя аж в трех местах (тут name="имя", тут isset($_POST['имя']) и тут echo $_POST['имя']).
Мы можем также захотеть сменить имя какого-либо инпута и опять-таки это придется делать в трех местах.
Для того, чтобы поправить проблему, давайте сделаем PHP функцию для генерации инпута (назовем ее input), которая параметром будет принимать имя инпута, а возвращать его HTML код.

Чтобы была понятна цель, которой мы хотим достигнуть, сразу покажу, как мы будем пользоваться нашей функцией input. Давайте сделаем форму, а в ней один инпут с именем age. Сама форма будет представлять собой HTML код, а инпут будет создаваться PHP-шной вставкой:

```
<form action="" method="POST">
    <?php echo input('age'); ?>
</form>
```

В результате в браузер отправится следующий HTML со сгенерированным нашей PHP функцией инпутом:

```
<form action="" method="POST">
    <input name="age">
</form>
```

Давайте напишем реализацию функции input на PHP, пока без сохранения значения инпута после отправки. В этом случае код будет очень простой:

```
<?php
    function input($name)
    {
        return '<input name="' . $name . '">';
    }
?>
```

Пусть у нас дана вот такая HTML форма:

```
<form action="" method="POST">
    <input name="name">
    <input name="age>
    <input name="salary">
    <input type="submit" value="добавить работника">
</form>
```

Давайте перепишем ее с помощью нашей функции input:

```
<form action="" method="POST">
```

```
        <?php echo input('name'); ?>
        <?php echo input('age'); ?>
        <?php echo input('salary'); ?>
        <input type="submit" value="добавить работника">
</form>
```

Резюмируем: теперь получается, что для создания инпутов мы не пишем HTML код напрямую, а пишем PHP код, который результатом своей работы выдаст нужный нам HTML.
Зачем мы это делаем: затем, что в данном случае PHP код для инпута проще и нет не нужного нам дублирования имени инпута в трех местах.
Итак, думаю, основная идея введения таких функций вам понятна.
Давайте теперь допишем нашу функцию так, чтобы она сохраняла значение инпута после отправки.
Для этого в атрибут value генерируемого инпута будем подставлять данные из массива $_POST.
При этом для инпута с именем age мы подставим значение из $_POST['age'], для инпута с именем name - $_POST['name'] и так далее.
Так как у нас одна функция создает инпуты с разными именами и имя инпута хранится в переменной $name, то получится что value каждого инпута хранится в $_POST[$name].
То есть получим следующую функцию:

```
<?php
    function input($name)
    {
        $value = $_POST[$name];
        return '<input name="' . $name . '" value="' . $value .'">';
    }
?>
```

Наша функция, однако, не идеальна - ведь форма может быть еще не отправлена и $_POST будет пустой.
Поправим проблему, добавив условие if:

```
<?php
    function input($name)
    {
        if (isset($_POST[$name])) {
```

```
                $value = $_POST[$name];
            } else {
                $value = '';
            }

            return '<input name="' . $name . '" value="' . $value .'">';
        }
    ?>
```

Теперь наша функция работает так, как надо.
Давайте соберем вместе HTML форму и PHP код для вставки работника в базу данных:

```
<?php
    function input($name)
    {
        if (isset($_POST[$name])) {
            $value = $_POST[$name];
        } else {
            $value = '';
        }

        return '<input name="' . $name . '" value="' . $value .'">';
    }

    if (!empty($_POST)) {
        $name = $_POST['name'];
        $age = $_POST['age'];
        $salary = $_POST['salary'];
```

```
        $query = "INSERT INTO workers SET name='$name', age='$age', salary='$salary'";
        mysqli_query($link, $query) or die(mysqli_error($link));
    }
?>
<form action="" method="POST">
    <?php echo input('name'); ?>
    <?php echo input('age'); ?>
    <?php echo input('salary'); ?>
    <input type="submit" value="добавить работника">
</form>
```

Если вы запустите этот код и нажмете на кнопку - форма отправится, работник добавится в БД, но введенные данные из инпутов не исчезнут.
Если нажать на кнопку второй раз - в базу добавится такая же запись.
Можно также подредактировать в инпуте, к примеру, фамилию работника, оставив остальное без изменения и опять нажать на кнопку сохранения. В этом случае добавится второй работник с такими же данными, но с другой фамилией.
То есть удобство тут в том, что при массовой вставке те данные, которые не изменяются - можно не вбивать повторно.
Кроме того, то, что инпуты не очищаются сами - удобно для контроля вбитых нами данных после отправки формы.