Лабораторная работа № 5

Обработка форм. Работа с файлами.

Изучить создание и обработку форм.

Изучить основы работы с файлами.

Формы являются важнейшим средством взаимодействия веб-приложения с пользователем: с их помощью осуществляется ввод данных, выбор значений, авторизация и многое другое.

Тег	Основные атрибуты	Пример

`<form></form>`	action method	`<form action="test.php" method="post">`
		…
		`</form>`

`<label></label>`	for	`<input id="inp"/>`
		`<label for="inp">Текст</label>`
		или
		`<label><input type="..."/>`
		`Текст</label>`

`<input type="text"/>`	value size placeholder name	`<input type="text" size="20" name="t1" placeholder="Введите имя"/>`
`<input type="checkbox"/>`	value checked	`<input type="checkbox" value="0" checked="checked" />` `<input type="checkbox" checked value="0">`
`<input type="radio"/>`	value checked name	`<input type="radio" value="0" name="r1">` `<input type="radio" value="1" name="r1" checked>`
`<input type="button"/>`	value	`<input type="button" value="Кнопка">`
`<input type="submit"/>`	value	`<input type="submit" value="Отправить">`
`<input type="reset"/>`	value	`<input type="reset" value="Сброс">`
`<input type="hidden"/>`	value	`<input type="hidden" value="0" id="hid">`
`<textarea></textarea>`	name rows cols	`<textarea rows="10" cols="45" name="text"></textarea>`

`<select>`
`<option></option>`
`</select>`
	value	`<select id="s1">`
		`<option value="0" selected>Выберите город</a>`
		`<option value="1">Москва</a>`
		`<option value="2">Нью-Йорк</a>`
		`</select>`

Пример 1 – Калькулятор

```
<!doctype html>
<html>

<head>
    <title>Пример 1</title>
    <meta charset="utf8">
    <style>
        form>label {
            display: block;
            margin: 5px 10px
        }

        input {
            padding: 2px;
        }

        fieldset {
            width: 150px;
            margin: 5px 10px
        }

        input[type="submit"] {
            border: 1px solid #a32500;
            background: #efe4bd;
            margin: 5px 10px;
            padding: 4px;
        }
    </style>
</head>

<body>
    <form action="calc.php" method="post">
        <label>Операнд 1
            <input type="text" placeholder="Введите первое число" name="op1">
        </label>
        <label>Операнд 2
            <input type="text" placeholder="Введите второе число" name="op2">
        </label>
        <fieldset>
            <legend>Вид операции</legend>
            <label>+<input type="radio" name="r1" value="+" checked></label>
            <label>-<input type="radio" name="r1" value="-"></label>
            <label>*<input type="radio" name="r1" value="*"></label>
            <label>/<input type="radio" name="r1" value="/"></label>
        </fieldset>
        <input type="submit" value="Вычислить">
    </form>
</body>

</html>
```

Общие принципы обработки форм на PHP сводятся к методам получения данных из

объектов формы. Стандартным способом получения данных из формы является использование суперглобальных массивов, через которые можно получить доступ к данным формы: **$_POST** – содержит все POST-данные; **$_GET** – используется, если данные передавались через адресную строку; **$_REQUEST** – универсальный, содержащий данные, передаваемые методом POST, GET и данные COOKIE. Используется, если источник данных не имеет значения.

Пример кода для получения данных из формы:

```
<?
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    if (isset(($_POST['name'])) {
        echo htmlspecialchars($_POST['name']); // name соответствует имени
                                               элемента управления
    }
}
?>
```

Пример 2 – Обработчик формы калькулятора

```
<!doctype html>
<html>

<head>
    <title>Калькулятор</title>
    <meta charset="utf8">
    <style>
        p {
            font-size: 1.3em;
        }

        .error {
            color: red;
            font-size: 1.5em;
        }
    </style>
</head>

<body>
    <?php
    if (isset($_POST["op1"]) && isset($_POST["op2"])) {
        switch ($_POST["r1"]) {
            case "+":
                $r = $_POST["op1"] + $_POST["op2"];
                break;
            case "-":
                $r = $_POST["op1"] - $_POST["op2"];
                break;
            default:
                $r = "Операция не поддерживается";
        }
        echo "<p>Результат: $r</p>";
    } else {
```

```
        echo <<<EOD
 <p class="error">Не все операнды определены</p>
EOD;
    }
    ?>
</body>

</html>
```

Часто бывает необходимо реализовать форму и обработчик в одном файле. В этом случае нужно проверить, существует ли соответствующее значение в массиве данных. Если оно есть, то производится обработка, если нет, показывается форма. Для такого «проверочного» значения можно использовать скрытое поле (с атрибутом hidden), либо проверять, получены ли данные соответствующим методом (пример 3).

Пример 3 – Калькулятор (версия 2), форма и обработчик в одном файле

```
<!doctype html>
<html>

<head>
    <title>Калькулятор</title>
    <meta charset="utf8">
    <style>
        p {
            font-size: 1.3em;
        }

        .error {
            color: red;
            font-size: 1.5em;
        }

        form>label {
            display: block;
            margin: 5px 10px
        }

        input {
            padding: 2px;
        }

        fieldset {
            width: 150px;
            margin: 5px 10px
        }

        input[type="submit"] {
            border: 1px solid #a32500;
            background: #efe4bd;
            margin: 5px 10px;
            padding: 4px;
        }
    </style>
```

```
</head>

<body>
    <?php
    // проверим, есть ли данные
    if ($_SERVER['REQUEST_METHOD'] != 'POST') {
        //данных нет, показываем форму
        ?>
        <form action="calc2.php" method="post">
            <label>Операнд 1 <input type="text" placeholder="Введите первое число"
name="op1"></label>
            <label>Операнд 2 <input type="text" placeholder="Введите второе число"
name="op2"></label>
            <fieldset>
                <legend>Вид операции</legend>
                <select name="s1">
                    <option value="+">+</option>
                    <option value="-">-</option>
                    <option value="*">*</option>
                    <option value="/">/</option>
            </fieldset>
            <input type="submit" value="Вычислить">
        </form>
        <?php
    }
    // данные есть, проверяем остальное и вычисляем
    else {
        if (!empty($_POST["op1"]) && !empty($_POST["op2"])) {
            switch ($_POST["s1"]) {
                case "+":
                    $r = $_POST["op1"] + $_POST["op2"];
                    break;
                case "-":
                    $r = $_POST["op1"] - $_POST["op2"];
                    break;
                default:
                    $r = "Операция не поддерживается";
            }
            echo "<p>Результат: $r</p>";
        } else {
            echo <<<EOD
 <p class="error">Не все операнды определены</p>
EOD;
        }
    }
    ?>
</body>

</html>
```

Для работы с файлами в php используются функции, реализующие стандартные файловые операции.

Функция	Действие, пример

resource **fopen** (string $filename , st ring $mode [, bool $use_include_path = false [, resource$context]])
 	
Режимы:
'r' для чтения; помещает указатель в начало файла.
'r+' для чтения и записи; помещает указатель в начало файла.
'w' для записи; помещает указатель в начало файла и обрезает файл до нулевой длины. Если файл не существует - пробует его создать.
'w+' для чтения и записи; помещает указатель в начало файла и обрезает файл до нулевой длины. Если файл не существует - пытается его создать.
'a' для записи; помещает указатель в конец файла. Если файл не существует - пытается его создать.
'a+' для чтения и записи; помещает указатель в конец файла. Если файл не существует - пытается его создать.
	Открывает файл или URL.
	`//открыть файл для чтения`
	`$handle = fopen("test.txt", "r");`
	`//открыть файл для дозаписи, если его нет - создать`
	`$handle = fopen("test.txt", "a+");`

string **fread** (resource $handle , int $length)	Бинарное чтение.
	`<?php`
	`// получает содержимое файла в строку и выводит ее`
	`$filename = "test.txt";`
	`$handle = fopen($filename, "r");`
	`$contents = fread($handle, filesize($filename));`
	`echo $contents; fclose($handle);`
	`?>`

int **fwrite** (resource $handle , string $string [, int $length]) Возвращает количество записанных байт или FALSE.	Бинарная запись. Пример 4.
bool **fclose** (resource $handle)	Закрывает ранее открытый файл. Пример 4.

bool **file_exists**(string $filename)	Проверяет наличие файла.
	`<?php`
	`$filename = '/path/to/foo.txt';`
	
	`if (file_exists($filename)) {`
	`    echo "Файл $filename существует";`
	`} else {`
	`    echo "Файл $filename не существует";`
	`}`
	`?>`

bool **is_writable**(string $filename)	Возвращает TRUE, если файл filename существует и доступен для записи. Пример 4.
string **file_get_contents** (string $filename [, bool $use_include_path = false [, resource$context [, int $o ffset = -1 [, int $maxlen]]]])	Читает файл в строку. Пример 5.
int **file_put_contents** (string $fi lename , mixed $data [, int $flags = 0 [, resource $context]]) Флаги: **FILE_USE_INCLUDE_PATH:** ищет filename в подключаемых директориях. **FILE_APPEND:** если файл filename уже существует, данные будут дописаны в конец файла вместо того, чтобы его перезаписать. **LOCK_EX:** получить эксклюзивную блокировку на файл на время записи.	Пишет строку в файл. Если filename не существует, файл будет создан. Иначе, существующий файл будет перезаписан, за исключением случая, если указан флаг **FILE_APPEND**. Пример 5.
string realpath (string $path)	Возвращает канонизированный абсолютный путь к файлу. Пример 5.

Пример 4. Бинарная запись

```php
<?php
$filename = 'test.txt';
$somecontent = "Добавить эту строку к файлу\n";
//  файл существует и доступен для записи? if (is_writable($filename)) {

// $filename в режиме "записи в конец".

if (!$handle = fopen($filename, 'a')) {
    echo "Не могу открыть файл ($filename)";
    exit;



    // Записываем $somecontent в файл.
    if (fwrite($handle, $somecontent) === FALSE) {
        echo "Не могу произвести запись в файл ($filename)";
        exit;
    }

    echo "Записано ($somecontent) в файл ($filename)";

    fclose($handle);
} else {
    echo "Файл $filename недоступен для записи";
}
?>
```

Пример 5. Запись и чтение строк

```php
<?php

$file = 'test_strings.txt';

// Новые данные, которые нужно добавить в файл define("divider", "|");
$logdate = date("d.m.y G:i:s");

$serviceNumber = 898820;

$person = "John Smith";

$log = $logdate . divider . $serviceNumber . divider . $person . "\n";

/* Пишем содержимое в файл, используя флаг FILE_APPEND flag для дописывания
содержимого в конец файла
И флаг LOCK_EX для предотвращения записи данного файла кем-нибудь другим в данное
время */

if ($bytes = file_put_contents($file, $log, FILE_APPEND | LOCK_EX))
    echo "Успешная запись $bytes байт";
$fullname = realpath($file);
echo "<p>Файл $fullname содержит данные</p>";
if ($content = file_get_contents($file))
    echo "<p>Успешное чтение " . strlen($content) . " байт</p>";
echo "<div>$content</div>";
?>
```

Задания.

1. Доработать калькулятор из примера 3 так, чтобы он выполнял все 4 операции.
2. Реализовать калькулятор (вер. 3), дополнительно к основным операциям также:
    1. вычисляющий квадратный корень (из операнда 1 или 2, выбор реализовать радиокнопкой);
    2. округляющий результат до 3 знаков после запятой* (реализовать чекбоксом).
3. Реализовать функционал генератора .htpasswd. Исходные данные: логин и пароль. При нажатии на кнопку «Создать» генерируется строка вида: логин:хэш_пароля**.
4. Доработать пример 5 для построчного вывода (каждую строку заключать в тег p).
5. Реализовать запись в файл данных формы (ФИО – текстовые поля, факультет – выбор из списка, пол (радиокнопки)). Набор данных записать в файл, находящийся в папке /lab5/text. Вывести данные из файла в таблице (заголовки полей соответствуют полям формы).
6. Дополнить функционал калькулятора (вер. 4) записью в файл calc.log всех операций. Содержимое файла выводить после формы калькулятора. Каждая строка в формате ДД.ММ.ГГГГ ЧЧ:ММ:СС **операнд1 действие операнд2 = результат** округление: **да/нет**. Пример строк вывода:

02.10.2023 14:55:15 **100 + 120 = 220** округление: **нет**

02.10.2023 14:55:15 **100 / 120 = 0.833** округление: **да**

Предусмотреть разный фоновый цвет для каждого вида операции из палитр flatuicolor.com.

7. Реализовать задачу 4 (лабораторная работа №4) с использованием формы ввода данных (текстовое поле – для ввода строки, кнопка «Определить цифры»).
8. Реализовать задачу 5 (лабораторная работа №4) с использованием формы ввода данных (текстовое поле – для ввода даты и времени, кнопка «Проверить»).
9. Реализовать задачу 6 (лабораторная работа №4) с использованием формы ввода данных (текстовое поле – для ввода пароля, кнопка «Проверить сложность»).

* round (x,3)

** crypt (s, base64_encode(s))