Использование форм для передачи данных в программу PHP. Тест математических знаний.

Лабораторная работа № А-6.

ЦЕЛЬ РАБОТЫ

Закрепление базовых знаний использования *PHP*; приобретение навыков работы с формами, как с основным инструментом получения входных данных для ВЕБ-приложений. Обзор возможностей языка *PHP* для ввода/вывода данных, в том числе с помощью почтовых сообщений.

ПРОДОЛЖИТЕЛЬНОСТЬ

2 академических часа (1 занятие)

РЕЗУЛЬТАТ РАБОТЫ

Размещенный на Веб-сервере и доступный по протоколу *http* документ (одна страница сайта) с формой, позволяющей определить математическую задачу и ввести предполагаемый ответ. При отправке формы осуществляется автоматическое решение задачи, сравнение переданного и полученного результата, вывод и при необходимости отправка по электронной почте результатов вычисления и сравнения.

ДОПОЛНИТЕЛЬНЫЕ ТРАБОВАНИЯ К РАБОТЕ

Результата лабораторной работы – одна *html*-страница: вывод формы, обработка данных и вывод выполняется одной *PHP*-программой. При первой загрузке на странице сайта должна отображаться форма, содержащая следующие элементы.

- Кнопка "Проверить".
- Однострочные поля для ввода текста:
  - ФИО;
  - номер группы;
  - значение A;
  - значение B;
  - значение C;
  - Ваш ответ;
  - Ваш е-майл.
- Многострочные поля для ввода текста:
  - немного о себе.
- Селектор со следующими опциями:
  - площадь треугольника;
  - периметр треугольника;
  - объем параллелепипеда;
  - среднее арифметическое;
  - другие, самостоятельно придуманные опции (общее число опций – не менее 6).
- Флажок:
  - отправить результат теста по е-майл.
- Селектор со следующими опциями:
  - версия для просмотра в браузере;
  - версия для печати.

Математическая задача определяется с помощью селектора. Входными данными для нее являются три введенных значения: *A*, *B* и *C*. Например, если выбрана задача среднее арифметическое, то ее решением является (*A*+*B*+*C*)/3. При загрузке страницы в качестве значений берутся и выводятся как начальные значения полей произвольные числа от 0 до 100. В качестве

значений могут выступать как целые числа, так и десятичные дроби (допустимо использовать в дробях как точку, так и запятую).

Флажок "Отправить результат теста на е-майл" изначально не отмечен; поле "Е-майл" вместе с надписью скрыто. При отметке флажка поле и соответствующая надпись появляются в форме (реализуется с помощью JavaScript).

Все элементы формы должны быть сгруппированы так, чтобы их было удобно использовать. Все они должны быть выровнены (включая кнопку) по левому краю, быть одинакового размера (кроме флажка и кнопки). Подписи к элементам должны располагаться слева от них.

При нажатии кнопки "Проверить" страница перезагружается с тем-же URL что и прежде. В зависимости от выбранного в списке значения внешний вид страницы может быть простым (для печати) или более сложным (для просмотра в браузере). На странице выводятся отчет со следующими данными:

- ФИО и группа студента;
- сведения о студенте (немного о себе);
- тип задачи;
- входные данные (числа);
- предполагаемый тестируемым результат (вычисленный человеком результат решения задачи);
- вычисленный программой результат;
- выводы об успешности теста ("Тест пройден" или "Ошибка: тест не пройден").

В случае, если был установлен флажок "Отправить результат теста по е-майл" – должна быть выведена надпись: "Результаты теста были автоматически отправлены на e-mail" с указанием введенного тестируемым адреса.

Если был выбран пункт списка "Версия для просмотра в браузере", то внизу результатов теста должна выводиться ссылка "Повторить тест", при переходе по которой должна вновь загрузиться форма (произвольные числа в полях *A*, *B* и *C* должны быть другими). Поля "*ФИО*" и "*Номер группы*" должны быть заполнены ранее введенными значениями. Кнопка оформляется в виде ссылки `<a>` (с фоном и рамкой), реагирует на курсор мыши.

РЕКОМЕНДАЦИИ К СТРУКТУРЕ ПРОГРАММЫ

Представленное в лабораторной работе задание является классическим обработчиком данных формы. Причем, исходя из задания, формировать элементы формы и выводить результаты ее обработки должна одна программа.

Рассмотрим сначала обработку данных. Обычно обработчики формы располагаются в самом начале html-страницы, т.к. от результата их работы зачастую зависит ее содержание.

Листинг А-6. 1

```
<?php
if( isset( $_POST['A'] ) ) // если из формы были переданы данные
{
      ... // обработчик переданных из формы данных
}
?><!doctype html>
...
```

Явным и единственным признаком того, что пользователем была заполнена форма и ее данные переданы в скрипт для обработки – это наличие элементов с именами тегов формы в массиве `$_POST` или `$_GET` (в зависимости от метода передачи данных). Т.к. в лабораторной из формы передается большой массив данных (включая многострочный текст), то разумно использовать *POST*-форму. Следовательно, ее данные будут размещены в массиве `$_POST`. Зная имена элементов формы, можно проверить наличие в массиве элементов с соответствующими ключами. Если они присутствуют – они были переданы из формы (им просто неоткуда взяться в массиве, кроме как быть переданными из формы), а значит пользователь заполнил ее и отправил для обработки, нажав кнопку "Ок".

В данном случае достаточно проверить наличие в массиве только одного элемента с индексом "*A*", т.к. страница обрабатывает только одну форму. Если в обработчик могут быть переданы данные из нескольких форм, необходимо так подбирать имена их элементов, чтобы можно было однозначно определить из какой формы были отправлены данные. Часто для этого используют невидимые текстовые поля в которых записывают имя формы или какой-либо идентификатор.

Листинг А-6. 2

```
<form name="form_1" method="post" action="/">
      <input type="hidden" name="PROCESS" value="1">
</form>

<form name="form_2" method="post" action="/">
      <input type="hidden" name="PROCESS" value="2">
</form>
```

Итак, наличие в массиве `$_POST` элемента с ключом "*A*" говорит о том, что форма была заполнена, данные переданы в программу, а значит в массиве присутствуют и другие элементы, соответствующие полям формы. Теперь их необходимо обработать.

Листинг А-6. 3

```
if( isset( $_POST['A'] ) ) // если из формы были переданы данные
{
     if( $_POST['TASK'] == 'mean' )  // если вычисляется среднее арифметическое
     {
          $result = round( ($_POST['A']+$_POST['B']+$_POST['C'])/3, 2 );
     }
     else
     if( $_POST['TASK'] == 'perimetr' )   // если вычисляется периметр
     {
          $result = $_POST['A']+$_POST['B']+$_POST['C'];
     }
}
```

Первым шагом обработки является автоматическое решение математической задачи. Тип задачи, т.е. алгоритм вычисления результата, зависит от выбранной опции в соответствующем селекторе. Исходя из анализа переданного значения селектора с помощью ряда условных операторов, задача успешно решается выбранным алгоритмом (одним или рядом арифметических действий), а результат сохраняется в переменной `$result`. В некоторых случаях разумно округлить результата с помощью функций PHP, в данном примере это функция `round()` с округлением до двух знаков после запятой. Самостоятельно добавьте в код обработку решения других математических задач.

Обратите внимание – переменная инициируется только при решении задачи, а значит если она определена, то в программе ниже это также является признаком передачи данных из формы. Исходя из этого выводится результат обработки данных.

Причем результатом обработки может являться и сама форма! Действительно, при ошибке обработки необходимо повторно запросить данные. Или же, как в данной лабораторной работе, отсутствие данных в массиве `$_POST` означает что форма не была заполнена, а значит ее необходимо вывести заново. Или можно сказать, что результатом обработки пустых входных данных является их повторный запрос, т.е. вывод формы.

Листинг А-6. 4

```
if( isset( $result )       // если форма была обработана
{
     ...              // выводим результаты обработки
}
else                  // если форма не обработана (данные не переданы в PHP)
{
     echo '<form name="form" method="post" action="/">';  // выводим форму
     ...              // выводим теги элементов формы
```

```
     echo '</form>';
}
```

Теперь, зная результат решения задачи и все переданные из формы данные, можно довольно просто вывести требуемый отчет.

Листинг А-6. 5

```
echo 'ФИО: '.$_POST['FIO'].'<br>';   // выводим ФИО студента
echo 'Группа: '.$_POST['GROUP'].'<br>';   // выводим группу студента

if($_POST['ABOUT'] ) // если сведения "немного о себе" заполнены
      echo '<br>'.$_POST['ABOUT'].'<br>'; // выводим их

echo 'Решаемая задача: ';       // выводим тип решаемой задачи
if( $_POST['TASK'] == 'mean' ) echo 'СРЕДНЕЕ АРИФМЕТИЧЕСКОЕ'; else
if( $_POST['TASK'] == 'perimetr' ) echo 'ПЕРИМЕТР ТРЕУГОЛЬНИКА';

if( $result === $_POST['result'] )   // если вычисления человека и машины совпали
     echo '<br><b>ТЕСТ ПРОЕДЕН</b><br>';  // выводим поздавления
else                                 // если человек ошибся
     echo '<br><b>ОШИБКА: ТЕСТ НЕ ПРОЙДЕН!</b><br>'; // ругаемся
```

Программа формирует отчет и непосредственно в процессе его формирования выводит его в HTML-код браузера. Это вполне допустимо если формируемый текст нигде более не используется, но, по условиям лабораторной работы, он должен быть отправлен по электронной почте при установке соответствующего флажка. Т.е. результат используется дважды: для отправки по почте и для вывода в браузер. Поэтому сохраним отчет в промежуточно переменной `$out_text`, которую затем и будем использовать необходимое число раз.

Листинг А-6. 6

```
$out_text='ФИО: '.$_POST['FIO'].'<br>';   // подготавливаем содержимое отчета
$out_text.='Группа: '.$_POST['GROUP'].'<br>';

if($_POST['ABOUT'] ) $out_text.='<br>'.$_POST['ABOUT'].'<br>';

$out_text.='Решаемая задача: ';
if( $_POST['TASK'] == 'mean' ) $out_text.='СРЕДНЕЕ АРИФМЕТИЧЕСКОЕ'; else
if( $_POST['TASK'] == 'perimetr' ) $out_text.='ПЕРИМЕТР ТРЕУГОЛЬНИКА';

if($result === $_POST['result']) $out_text.='<br><b>ТЕСТ ПРОЕДЕН</b><br>'; else
                          $out_text.='<br><b>ОШИБКА: ТЕСТ НЕ ПРОЙДЕН!</b><br>';

echo $out_text;      // выводим отчет в браузер

if( array_key_exists('send_mail', $_POST) )     // если нужно отправить результаты
{
     // отправляем результаты по почте простым письмом
     mail( $_POST['MAIL'], 'Результат тестирования',
            str_replace('<br>', "\r\n", $out_text),
            "From: auto@mami.ru\n"."Content-Type: text/plain; charset=utf-8\n" );

     // выводим соответствующее сообщение в браузер
     echo 'Результаты теста были автоматически отправлены на e-mail '.$_POST['MAIL'];
}
```

В представленной программе отчет сначала формируется, а выводится в браузер только после полного завершения этого процесса. Для отправки письма с отчетом необходимо проверить установку соответствующего флага. Напомним, что если флаг установлен, то он передается в обработчик формы, если нет – то не передается. Поэтому признаком установки флага является наличие элемента массива `$_POST` с совпадающим с именем флага ключом. В программе проверяется это условие и, при его выполнении, формируется и отправляется письмо.

В примере письмо отправляется обычным текстом, а не *HTML*-кодом, поэтому необходимо заменить все теги перевода строки `<br>` на соответствующие символы (символы "Возврат каретки" и "Перевод строки"). После отправки выводится требуемая надпись.

Для завершения основной части PHP кода также необходимо сформировать *HTML*-код кнопки, при нажатии которой будет повторно отображена форма, а тестирование можно будет провести заново.

Листинг А-6. 7

```
echo '<a href="?F='.$_POST['GROUP'].'&G='.$_POST['GROUP'].
     '" id="back_button">Повторить тест</a>';
```

Кнопка формируется в виде ссылки, оформление которой осуществляется в CSS-файле. Т.к. после перехода по ссылке будет проводиться повторное тестирование, то, согласно условиям лабораторной работы, поля формы с ФИО и группой студента уже должны быть заполнены: для этого в адрес ссылки передаются соответствующие GET-параметры. При выводе самой формы довольно просто проверить существование указанных ключей в массиве `$_GET` и, если они там есть, вывести элементы массива как значения полей.

Самостоятельно дополните программу обработкой других видов задач, дополнением отчета входными данными, вычисленным и переданным результатом вычисления. При этом, если предполагаемый результат не был передан (пустая строка) – должна формироваться надпись: "Задача самостоятельно решена не была".

Самостоятельно разделите способ предоставления данных "Для отображения в браузере" и "Для печати" так, чтобы они были одинаковы по содержанию, но разные по внешнему виду. Кнопку "Повторить тест" необходимо выводить только в первом случае.

Самостоятельно сформируйте форму и назовите ее элементы в соответствии с используемыми в рассмотренных выше примерах ключами массива `$_POST`. Дополните форму соответствующим кодом *JavaScript* для выполнения на стороне клиента (*JavaScript* выводится методами *PHP*, как и любой другой текст *HTML*-кода страницы). Добавьте в код формы вывод произвольных чисел как начальные значения полей *A*, *B* и *C*, а также вывод ФИО и группы студента, если страница была сформирована повторно.

СПРАВОЧНАЯ ИНФОРМАЦИЯ

Получение произвольного числа	`$some_val = mt_rand(5,100);`
	Функция `mt_rand()` возвращает случайное целое число в указанном диапазоне (в данном примере от 5 до 100). Если необходимо получить случайное вещественное число, можно использовать следующий код PHP:
	`$some_val =  mt_rand(500,10000)/100;`

Замена подстроки в строке	`$s = str_replace ( $search, $replace, $str );` Функция заменяет в строке `$str` все подстроки `$search` на строку `$replace`. Подробнее о функции – см. в справочнике.
Получение данных о переданных в документ GET- и POST-параметрах.	`$_GET['el_name']` – значение элемента *GET*-формы с именем "*el_name*". Такие параметры также можно передавать и через *URL*. `$_POST['el_name']` – значение элемента *POST*-формы с именем "*el_name*".
Отправка сообщения по электронной почте	`mail($adress, $subject, $message, $headers);` Отправляет по адресу `$adress` сообщение с темой `$subject` и текстом `$message`. Заголовки `$headers` указывают как именно письмо должно отображаться в почтовой программе, могут использоваться и для других целей. Подробнее о функции – см. в справочнике.
Скрыть и показать элемент в	`<input type="checkbox" name="send_mail" onClick="`

зависимости от состояния флажка (код JavaScript)	`obj=document.getElementById('object');`
	`if( this.checked )`
	`  obj.style.display='block';`
	`else`
	`  obj.style.display='none';">`
	
	`<div id="object">...</div>`
	Непосредственно в теге флажка определяем обработчик события нажатия на кнопку мыши. В событии проверяем установлен ли флажок. Если да, то отображаем объект с помощью изменения стиля. Если нет – то скрываем его.

КОНТРОЛЬНЫЕ ВОПРОСЫ К ЛАБОРАТОРНОЙ РАБОТЕ

Для успешной защиты работы помимо соответствующего требованиям результата необходимо уверенно отвечать на нижеперечисленные и другие вопросы, а также на контрольные вопросы всех предыдущих лабораторных работ.

1. Что такое форма?
2. Какие типы форм бывают?
3. В чем отличие POST- и GET-форм?
4. Какие элементы формы Вы знаете?
5. Как передает свое значение каждый элемент формы в PHP-программу?
6. Как округлить вещественное число до определенного количества разрядов после запятой?
7. Сколько символов может максимально содержать строка в которой буферизуется HTML-код?
8. Как отправить сообщение по электронной почте средствами PHP?
9. Почему в листинге А-6.5 при сравнении переданного и вычисленного результата используется оператор эквивалентности?
10. На какую функцию можно заменить array_key_existst() в листинге А6. 6?
11. Можно ли в листинге А6. 6 буферизовать сообщение об отправки письма в переменной $out_text? Если да – то как это сделать и будет ли такой код более оптимальным?
12. Какие события могут обрабатываться в JavaScript? Какими способами можно задать обработчик события для объекта?