|  |
| --- |
| **Технология web-программирования, группа АФТ 05-17, ОЗ АФТ 05-17** |
| Дата занятия | Вид занятия | Ссылка на источник |
| 13.04.2020 | Лк | 1. Видеолекция «PHP видео-урок №2. Как написать скрипт PHP программирование?»

https://www.youtube.com/watch?v=kpDDcx3Geqc1. Торопова О.А. Основы web-программирования. Технологии HTML, DHTML [Электронный ресурс]: учебное пособие / О.А. Торопова, И.Ф. Сытник. – Электрон. текстовые данные. – Саратов: Саратовский государственный технический университет имени Ю.А. Гагарина, ЭБС АСВ, 2012. – 106 c. – 978-5-7433-2606-8. – Режим доступа: http://www.iprbookshop.ru/76493.html
2. Одиночкина С.В. Web-программирование PHP [Электронный ресурс] / С.В. Одиночкина. – Электрон. текстовые данные. – СПб. : Университет ИТМО, 2012. – 79 c. – 2227-8397. – Режим доступа: http://www.iprbookshop.ru/65750.html
3. Электронный ресурс http://weblecture.ru/lecture?num=181
 |
| 13.04.2020 | Лб | Выполнить лабораторную работу № 4 «Взаимодействие PHP-сценариев с формой Web-страницы» Методические указания по выполнению лабораторной работы приведены ниже  |

**Лабораторная работа 4.**

**Взаимодействие PHP-сценариев с формой Web-страницы.**

**Цель работы:** научиться создавать основные элементы управления на форме Web-страницы и устанавливать связь со сценарием PHP.

**Поясняющая информация**

Одним из важнейших свойств web-документов является возможность **получения данных от клиентов** и **отправки их на сервер**. Эта возможность обеспечивается с помощью так назваемых **ФОРМ** (**FORMS**). Таким образом,

**Формы** - это фрагменты **HTML**-документов, "ответственные" за **ввод информации клиентом.**

Вот один из простейших примеров:



Это очень простая форма. Она содержит лишь **текстовые поля** и две кнопки. Первая кнопка (**RESET**) позволяет **"сбросить"** все введенные пользователем данные, вторая (**SUBMIT**)- непосредственно **отсылает данные на сервер**. Если забыть разместить в форме кнопку **SUBMIT**, то данные на сервер не будут отправлены. Поэтому в форме обязательно должна присутствовать кнопка **SUBMIT**, "ответственная" за отправку данных, введенных клиентом, на сервер.

**Формы** размещаются в контейнере **<form>...</form>**.

Этот контейнер можно уподобить обычному бумажному почтовому конверту - на последнем обязательно присутствует адрес назначения и фамилия адресата, в тэге **<form>...</form>** непременно указывается **адрес серверного сценария**, который получит и обработает ввведенные клиентом данные. Для этого используется атрибут **action**. Таким образом, простейшая форма имеет такую структуру:

|  |
| --- |
| **<form action="адрес серверного сценария">....Всякие элементы форм (среди которых непременно присутствует SUBMIT).....</form>**  |

Атрибут **action** является обязательным (если, конечно, отправка данных на сервер предусмотрена), но не единственным атрибутом формы. О других атрибутах мы поговорим чуть позднее, а пока остановимся на основных элементах форм.

Все элементы форм создаются тэгами, имеющими начало: **<input type="..."**, где значения атрибута **type** и определяют тип элемента. Рассмотрим возможные значения этого атрибута.

**Текстовое поле (text field):**

|  |
| --- |
| **<input type="Text" name="имя поля" value="начальный текст" align="LEFT" size="число символов" maxlength="максимальное число символов">**  |

 **name** - имя поля (впоследствии мы узнаем, как к элементам форм можно обращаться просто по имени);

 **value** - то, что будет записано в текстовое поле "по умолчанию", как только документ появится в окне браузера;

 **align** - этот атрибут "отвечает" за расположение текстового поля; он не является специфическим для элементов форм и (как и в других тэгах) может принимать значения: **left**, **right**, **center**;

 **size** - длина поля в символах;

 **maxlength** - максимальное число символов, которое можно ввести в поле. Если этот атрибут не указывать, ограничений на ввод не будет.

|  |
| --- |
| Например:код:**<input name="fio" value="Горбунков Семен Семенович" align="LEFT" size="28" maxlength="35">** результат:form2 |

Значением текстового поля должен быть только текст. Если, к примеру, поместить в него html-код, он будет отображаться как текст (а не так, как он выглядел бы в окне браузера). Но вот [пример размещения в текстовом поле бегущей строки](http://kek.ksu.ru/eos/tests/Primeri/pbegstrtxt.html), выполненный Сергеем Бурылиным. Как ему это удалось - попробуйте разобраться сами.

**Текстовая область (text area):**

Текстовые области удобно использовать для ввода большого количества текстовой информации.

Код текстовой области выглядит следующим образом:

|  |
| --- |
| **<textarea name="имя" cols="число столбцов" rows="число строк" wrap="способ переноса строк">ТЕКСТ, РАЗМЕЩЕННЫЙ В ТЕКСТОВОЙ ОБЛАСТИ</textarea>**  |

Здесь атрибут **"wrap"** указывает на способ разрыва строки: если присвоить ему значение **off**, вводимый текст будет **"уходить" за пределы области**, в противном случае (**on**, **soft** или даже просто **wrap** без присвоения) текст автоматически переносится на новую строку.

|  |
| --- |
| Например:код:**<TEXTAREA name=bio cols=30 rows=6 wrap>КРАТКАЯ АВТОБИОГРАФИЯ</TEXTAREA>** результат:form3 |

**Список:**
Списки позволяют сделать **единственный** или **множественный** выбор из набора предлагаемых **опций** (**options**).



Структура HTML-кода, определяющего список, такова:

|  |
| --- |
|  **<select name="имя списка" size="число видимых опций" multiple><option value="значение" selected>текст<option value="значение">текст...... и так далее......<option value="значение">текст<option value="значение">текст</select>** |

Коды списков, приведенных нами в качестве примеров, имеют вид:

|  |  |
| --- | --- |
| **<select name="week" size="7"><option value="1">Понедельник</option><option value="2">Вторник</option><option value="3">Среда</option><option value="4">Четверг</option><option value="5">Пятница</option><option value="6">Суббота</option><option value="7">Воскресенье</option></select>** Список с единственным выбором, показаны все имющиеся опции, ни одна не выделена.  | **<select name="food" size="4" multiple><option value="1" selected>Пицца</option><option value="2">Хот-дог</option><option value="3">Шашлык</option><option value="4">Мороженое</option><option value="5">Шампанское</option><option value="6">Ананас</option><option value="7">Шоколад</option><option value="8">Кофе</option></select>** Список с множественным выбором, показаны лишь 4 из 8-ми опций, первая опция выделена.  |

**"Флажок" (Checkbox)**

"Единичный" флажок создается с помощью кода:

|  |
| --- |
| **<input type="Checkbox" name="имя флажка" value="значение" checked> текст, который обычно помещается рядом с флажком** |

Атрибут **checked** указывает на то, что "флажок" по умолчанию будет отмеченным.
Часто используется сразу несколько "флажков", например:

|  |  |
| --- | --- |
| **Укажите Ваши навыки:<input type="checkbox" name="skill" value="1" checked>HTML<input type="checkbox" name="skill" value="2">JavaScript<input type="checkbox" name="skill" value="3">ASP<input type="checkbox" name="skill" value="4">Perl<input type="checkbox" name="skill" value="5">PHP<input type="checkbox" name="skill" value="6">XML<input type="checkbox" name="skill" value="7">FLASH** | Укажите Ваши навыки:form5 |

**"Радиокнопка" (Radiobutton)**

"Единичная" радиокнопка создается с помощью кода:

|  |
| --- |
| **<input type="radio" name="имя радиокнопки" value="значение" checked> текст, который обычно помещается рядом с радиокнопкой** |

Атрибут **checked** указывает на то, что радиокнопка по умолчанию будет отмеченной.
Часто используется сразу несколько радиокнопок, например:

|  |  |
| --- | --- |
| **Укажите Ваши навыки:<input type="radio" name="skill" value="1" checked>HTML<input type="radio" name="skill" value="2">JavaScript<input type="radio" name="skill" value="3">ASP<input type="radio" name="skill" value="4">Perl<input type="radio" name="skill" value="5">PHP<input type="radio" name="skill" value="6">XML<input type="radio" name="skill" value="7">FLASH** | Укажите Ваши навыки:form6 |
| Здесь все радиокнопки имеют **одно имя** - **skill**. Это обеспечивает **единственность** выбора. Если присвоить радиокнопкам разные имена, будет возможен **множественный выбор**, как и в случае использования **checkbox**'ов. |

**Кнопка (button):**

Каждая форма должна иметь кнопку, нажатие на которую обеспечивает **отправку данных на сервер**. Такая кнопка создается с помощью кода:

|  |
| --- |
| **<input type="submit" name="имя кнопки" value="то, что на этой кнопке написано">**  |
| Например: |
| **<input type="submit" name="subm" value="Отправить данные">**  |
| Результат:form7  |

При нажатии кнопки **submit** данные передаются на сервер по адресу указанному в качестве значения атрибута **action**. Чтобы предусмотреть возможность **сброса** всех введенных пользователем данных (и восстановления данных, указанных по умолчанию), используется кнопка **reset**. Код ее выглядит следующим образом:

|  |
| --- |
| **<input type="reset" name="имя кнопки" value="то, что на этой кнопке написано">**  |

Наряду с кнопками **submit** и **reset** используются и "простые" кнопки, с которыми обычно связывается то или иное действие. "Простая кнопка" создается с помощью кода:

|  |
| --- |
| **<input type="button" name="имя кнопки" value="то, что на этой кнопке написано" onclick="действие, связанное с кнопкой">**  |

Обратите внимание на атрибут **onclick**. Он представляет собой **обработчик события** - нажатия на кнопку. Для обработки этого события (и множества других событий) используются клиентские сценарии, написанные на языке **JavaScript** или **VBScript**. Как правило, эти сценарии оформляются в виде **функций**, вызов которых является значением обработчика события **onclick**.

|  |
| --- |
| Например: |
| **<input type="button" value="Показать сегодняшнее число" onclick="alert(Date());">**  |

Кнопки могут иметь и более "экзотический" вид. Так, на них можно разместить не только текст, но и рисунок, в том числе и анимированный, и вообще, **любой фрагмент HTML-кода**. Вот общий вид кода "нестандартных" кнопок:

|  |
| --- |
| **<button name="имя кнопки onclick="действие, связанное с кнопкой">HTML-код фрагмента, размещаемого на кнопке"></button>**  |

Кнопку с размещенной на ней картинкой можно создать и так:

|  |
| --- |
| **<input type="image" src="/адрес рисунка" onclick="действие, связанное с кнопкой">**  |

Однако в этом случае она мало напоминает кнопку. Сравните, например:

|  |  |
| --- | --- |
| htmlpencil00 | **<button style="width:124" onclick="alert('Это кнопка с картинкой');"><img src="/pictures/pencil.gif" mce\_src="/pictures/pencil.gif" width=118></button>**  |
|  | **<input type="image" src="/pictures/pencil.gif" onclick="alert('Это тоже кнопка с картинкой');">**  |

**Поле для ввода пароля (password):**

Поле для ввода пароля отличается от обычного текстового поля тем, что вводимое в него значение отображается с помощью звездочек. Синтаксис такого поля имеет вид:

|  |
| --- |
| **<input type="password" value="значение по умолчанию" size="длина поля (в символах)" maxlength="максимально допустимое число вводимых символов">**  |

Например:

|  |
| --- |
| Код:**<input type="password" value="123456" size="6" maxlength="8">**  |

### Атрибуты форм:

* **Action** - **адрес серверного сценария**, получающего и обрабатывающего данные, вводимы пользователем при помощи формы.
* **Name** - **имя формы**. Этот атрибут не является обязательным, но его наличие позволяет обращаться к форме (и ее отдельным элементам) "фамильярно", т.е. просто по имени.
* **Method** - **способ передачи данных**. Этот атрибут может принимать два значения: **GET** или **POST**. Подробнее об этих методах мы будем говорить позднее, когда будем изучать средства создания серверных сценариев, а пока скажем лишь только, что метод **GET** используется в тех случаях, когда объем передаваемых данных невелик (например, значение одной-двух переменных), если же передается большой объем данных (или его трудно оценить заранее), то применяется метод **POST**. И еще: если передается пароль, то метод **GET** использовать нельзя, так как в этом случае текст пароля будет "светиться" в адресной строке.
* **Enctype** - **тип передаваемых данных**. Если форма не содержит прикрепляемых файлов, этот атрибут можно не указывать (по умолчанию он примет значение: "**application/x-www-form-urlencoded**"); если же форма содержит поля типа **file**, значение этого атрибута должно быть "**multipart/form-data**".
Как правило, перед отправкой данных на сервер производится их предварительный анализ клиентскими средствами. Например, проверяется, заполнены ли обязательные поля, правилен ли формат адреса электронной почты (содержит знак '@') и пр. Для этого на языке **JavaScript** (или **VBScript**) создаются специальные сценарии.

**Взаимодействие формы с PHP-скриптом.**

Для ввода данных в PHP-скрипт из формы в самой форме необходимо предусмотреть имена полей и вызов PHP-скрипта из формы записью вида

**action="имя (и путь) php скрипта, обрабатывающего форму"**

***Пример:***

<form action="script\_1.php" method="GET"> *(или POST)*

<input type=text name='a'>

<input type=text name='b'>

<input type=submit value='Вызываем PHP'>

</form>

В вызываемом скрипте то, что вводилось в поля с именами (в примере **a** и **b**) будет присвоено одноименным переменным, если, конечно ими воспользуетесь. То есть **$a** и **$b** простейший script\_1.php, вызываемый формой, может выглядеть так:

<?php
$a=$\_GET['a'];

$b=$\_GET['b'];

print "В форме Вы ввели: <br> $a <br> $b";
?>

**Инструкция switch**

Инструкция switch позволяет разветвить действие в зависимости от значения выражения. Общий ее вид представлен далее:

 switch (выражение)
  {
   case значение\_1:
    действия, если выражение равно значению\_1
   break;

   case значение\_2:
    действия, если выражение равно значению\_2
   break;

   default:
    выполняется, если выражение не приняло не одно из значений
}

Пример применения инструкции switch приведен ниже. Обратите внимание, что в форме использовано поле типа select.

Итак - это **форма**, вызывающая скрипт:

Какая планета четвертая по счету от земли?
<form action="script2\_1.php" method="POST">
  <select name=n1 >
   <option value="Земля"> Земля
   <option value="Марс"> Марс
   <option value="Венера"> Венера
  </select>
  <input type=submit value="Готово">
</form>

А это **скрипт**, обрабатывающий форму.

<?php
  switch ($n1)
  {
    case "Земля":
      print "не верно";
    break;

    case "Марс":
      print "Верно !";
    break;

    case "Венера":
      print "не верно";
    break;
  }
?>

### Оператор ?

Оператор ? возвращает одно из двух значений, разделенный двоеточием. Первое - если выражение истинно, второе если ложно. То есть можно сказать, что ? (знак вопроса) - короткая альтернатива условному оператору if.

***Пример:***

Укажите Ваш пол
<form action="script2\_2.php" method="POST">
  <select name=n1 >
   <option value="М">Мужской
   <option value="Ж">Женский
  </select>
  <input type=submit value="Готово">
</form>

А это простенький **скрипт**, обрабатывающий форму.

<?php
  ($n1=="М")? print "Привет мужик": print "Здравствуйте дама";
?>

**Массивы**

Сперва просто термин: **Массив** - *это набор значений, скрытый под одним именем*. То есть если ранее мы пользовались переменными, у каждой из которых было свое имя (и хранила она одно значение), то благодаря массиву под одним именем можно хранить сразу много переменных. Для того, чтобы их все таки различать, помимо имени надо добавить индекс (в квадратных скобках).

Чтобы создать массив, и сразу поместить в него некоторые значения, необходима строка вида: **$имя\_массива = array (значения, которые войдут в массив).**

**Пример:**

<?php
  $user = array ("Василий", "Иван" , "Петр");

?>

Итак, мы занесли в массив 3 элемента. Они получили при этом свои номера, начиная с нуля. То есть $user[0] = "Василий" , $user[1] = "Иван" и $user[2] = "Петр". Соответственно, вывести элементы массива известной длины удобнее всего в цикле:

for ($i=0; $i<=2; $i++)

   { print "$user[$i] <br>";}

Элементы можно присваивать и не одной командой, а по мере необходимости:

  $user[] = "Василий";

  $user[]="Иван";

  $user[]="Петр";

**Работа с массивом.**

На практике, количество элементов массива мы часто не знаем. По этому весьма полезна функция, которая позволяет определить размер массива - **count ( $имя\_массива )**. Пример ее использования:

<?php
  $user = array ("Василий", "Иван" , "Петр", "Михаил", "Александр");

 print "Всего в массиве \$user - ". count($mas) . " элементов";

?>

Обратите внимание, что число элементов массива - 4, а номер последнего элемента - 3. Напоминаю, это связано с тем, что нумерация идет с нуля.

**Автоматический перебор элементов массива.**

Еще тонкость работы с массивами связана с тем, что номера элементов массива можно задать и "вручную", причем не обязательно подряд. Например:

  $user[5] = "Василий";
  $user[1]="Иван" ;
  $user[3]="Петр";

Перебрать все элементы такого массива "обычным" циклом не получится. Однако, в этом нет проблемы, поскольку PHP располагает удобными методами перебора любых массивов.

Перебрать все элементы массив позволяет конструкция вида:
   **foreach** (имя\_массива **as** переменная)
 {

переменная примет значение каждого элемента массива;

}
пример:

  $mas= array ("Иван","Петр","Василий");
  $mas[7] = "Михаил";
  $mas[5] = "Григорий";
  foreach ($mas as $a)
   {print " $a <br>";}

Данный скрипт выведет все элементы массива по порядку возрастания их номеров.

**Ассоциированный массив** - это массив, к элементу которого можно обратится по имени. (Иногда их называют структурами).

Создать ассоциированный массив можно следующим образом:

<?php
  Error\_Reporting(E\_ALL & ~E\_NOTICE);
  $mas = array (
   name1=>"Иван",
   city =>"Одесса",
   phone=>"322223");
?>

При этом будет создан массив с именем $mas и именами полей в нем name1, city и phone. В эти поля мы и заносим соответствующие данные о имени, городе и телефоне. Полностью аналогичный массив можно создать и **более кратким методом**:

<?php
  Error\_Reporting(E\_ALL & ~E\_NOTICE);
 $mas[name1]="Иван";
 $mas[сyti] ="Одесса";
 $mas[phone]="322223";
?>

**Вывести данные** из ассоциированного массива можно следующим образом:

<?php
  Error\_Reporting(E\_ALL & ~E\_NOTICE);
  print $mas[name1];
  print $mas[city];
  print $mas[phone];
?>

<

**Просмотреть в цикле** ассоциированный массив можно используя конструкцию –

**foreach** ($имя\_массива **as** $имена\_элементов => $значения)

   { переменные примут значение каждого имени и значения элемента массива; }

**Пример:**

<?php
  Error\_Reporting(E\_ALL & ~E\_NOTICE);
 $mas[name1]="Иван";
 $mas[сyti] ="Одесса";
 $mas[phone]="322223";

  foreach ($mas as $a=>$b)
   {print " $a - $b <br>";}
?>

**Объединение массивов.**

Объединить массивы в общий массив позволяет функция **array\_merge** (первый\_массив, второй\_массив). **Пример:**

<?php
 $mas1= array ("Иван","Петр","Василий");

 $mas2= array ("Аня","Лена","Света");

 $mas3= array\_merge ($mas1, $mas2);

  foreach ($mas3 as $a)

  {print " $a <br>";}

?>

В данном примере создаются 2 простых массива - $mas1 с мужскими именами и $mas2 с женскими. Далее эти массивы объединяются в единый массив $mas3 и выводятся командой foreach.

Аналогично можно объединять и ассоциированные массивы. Но тут есть "тонкость" - чтобы не потерять данные имена полей массивов не должны совпадать. В противном случае данные из первого массива будут заменены данными из одноименного поля второго (хотя в некоторых случаях это и необходимо).

**Пример:**

<?php
Error\_Reporting(E\_ALL & ~E\_NOTICE);

 $mas1[name1]="Иван";
 $mas1[сyti1] ="Одесса";

 $mas1[phone1]="322223";
 $mas2[name2]="Лена";
 $mas2[сyti2] ="Умань";

 $mas2[phone2]="223322";
  $mas3= array\_merge ($mas1, $mas2);

   foreach ($mas3 as $a=>$b)

    {print " $a - $b<br>";}

?>

***К сдаче лабораторной предоставляются:*** *работающие страницы на сервере.*

## Задания на лабораторные работы по использованию языка PHP

В данной лабораторной работе используется следующее программное обеспечение:

* web-сервер apache (<http://www.apache.org>);
* интепретатор языка PHP (<http://www.php.org>);
* СУБД mySQL (<http://www.mysql.com>);

**Внимание.** Любое полезное программное обеспечение можно найти по адресу <http://www.gnu.org>.

*В html документе создайте форму с текстовыми полями, в которые пользователь вводит информацию и кнопкой типа Submit. Далее после нажатия копки Submit вызывается php-скрипт. При написании тега формы по возможности использовать как можно больше элементов управления.*

1. В html документе создайте форму с текстовым полем, в которое пользователь вводит свой логин и кнопкой типа Submit. Далее после нажатия копки Submit вызывается php скрипт который проверяет, зарегистрирован ли этого пользователь. При этом таких пользователей (разных логинов должно быть 4). Если введен один из существующих логинов, должно выводится приветствие для этого человека. Например: введен логин Inan\_php , должно вывестись приветствие "Здравствуйте Иванов Иван Иванович". Если введен неизвестный логин должно вестись сообщение - "Вы не зарегистрированный пользователь!"
2. Написать html-форму для регистрации участников олимпиады по программированию, и после регистрации отправить участнику сообщение.
3. Написать html-форму для регистрации участников конкурса танцев, и после регистрации вывода информации об участниках этого конкурса.
4. Написать html-форму для регистрации на тематическом форуме по программированию, и после регистрации вывода информации по тематике форума.
5. Написать html-форму, обрабатывающую ввод анкетных данных клиента и реагирующую соответствующим образом. Например, программа может выводить строку «Здравствуйте, Иванов Петр Федорович! Спасибо за заказ! Мы рады приветствовать в вашем лице клиента нашей фирмы» или «Здравствуйте, Симонова Инна Леонидовна! Спасибо за заказ! Так как сумма вашего заказа превысила 5000 рублей, Вы получаете 10 %-ную скидку». Анкетные данные: ФИО, возраст, сумма заказа.
6. По выбору пользователя генерировать html-форму для ввода описания личности человека и отображать данные, введенные в эту форму.
7. Создать html-форму для приема сообщения посетителя и выводить его сообщение на другой странице.
8. Создать html-форму со списком товаров на продажу и выводить информацию о заказе посетителя на другой странице.
9. Создать html-форму со списком предоставляемых услуг вашей туристической фирмой. После выбора пользователем какой-нибудь услуги выводится более полная информация о ней.
10. Написать html-форму для заполнения анкеты-резюме при приеме на работу и выдачи сообщения.