Министерство образования и науки Республики Казахстан
Кокшетауский государственный университет им. Ш. Уалиханова
Факультет «Техники и технологии»
Кафедра «Информационных систем и вычислительной техники»

Мусабеков Ж.С.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторным работам по дисциплине

«Программирование на языке PHP»

для студентов специальности
5В070300 – Информационные системы
5В070400 – Вычислительная техника и
программное обеспечение

Форма обучения: **очная, заочная**

Ф.4.02-02

Министерство образования и науки Республики Казахстан
Кокшетауский государственный университет им. Ш. Уалиханова
Факультет «Техники и технологии»
Кафедра «Информационных систем и вычислительной техники»

УТВЕРЖДЕНО
Решением УМС
Председатель
_____Жакупова А.Д.
_____20___г.

Мусабеков Ж.С.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторным работам по дисциплине

Программирование на языке PHP

для студентов специальности
5В070300 – Информационные системы
5В070400 – Вычислительная техника и
программное обеспечение

Форма обучения: **очная, заочная**

Кокшетау 2015

Методические указания составлены Мусабековым Ж.С. на основании рабочего учебного плана и рабочей программы дисциплины «**Программирование на языке PHP**» для студентов по специальностям 5В070300 – Информационные системы, 5В070400 – Вычислительная техника и программное обеспечение и включает все необходимые сведения по выполнению тем лабораторных работ по дисциплине.

Рассмотрено на заседании кафедры
«Информационных систем и вычислительной техники»
« 08 » 01 2015 г. / Протокол № 6 /

Заведующий кафедрой Хан С. И.

Одобрено учебно-методической комиссией Факультета техники и технологии
« 14 » 01 2015 г. / Протокол № 3 /

Председатель УМК Булашева А.И.

Общие указания к лабораторным работам

Изучение дисциплины «Программирование на языке PHP» основано на сочетании теоретических, лабораторных занятий и самостоятельной работы студентов.

Особое место в изучении предмета занимают лабораторные работы, которые проводятся после изучения разделов дисциплины «Программирование на языке PHP», а также перед экзаменами по этому предмету. То есть лабораторные работы имеют цель закрепить, обобщить и систематизировать полученные знания теоретического материала на основе проведения лабораторных занятий.

Кроме того, на лабораторных работах студенты углубляют теоретические знания и овладевают навыками работы с техническими и инструментальными средствами и техникой проведения эксперимента. Даже при совершенном овладении теорией, но без умения проводить эксперимент, нельзя стать полноценным специалистом. Поэтому это умение важно выработать у студентов в ходе проведения лабораторных занятий.

Важнейшим условием получения хороших практических знаний является предварительная подготовка студентов к каждой лабораторной работе, а также понимание цели и содержания работы. Поэтому перед выполнением лабораторных работ студент должен изучить ее содержание и порядок выполнения, повторить основы теории по конкретной схеме эксперимента, связанного с выполнением данной работы.

Перед выполнением работ все студенты должны изучить правила техники безопасности применительно к работам за компьютером и соблюдать правила техники безопасности при работе за компьютером.

По окончании лабораторной работы каждый студент должен самостоятельно обработать результаты выполняемых им опытов и составить отчет о проделанной работе.

Правила техники безопасности при выполнении лабораторных работ

1. Общие требования безопасности.

Опасные производственные факторы: воздействие на человека электрического тока, электрического поля, рентгеновского излучения, ультрафиолетового излучения.

Действие факторов: вследствие неисправности кабеля, электрической розетки (вилки), замыкания в цепи пользователь компьютера попадает под напряжение.

2. Требования безопасности перед началом работы.

2.1. К работе на компьютере допускаются только лица, прошедшие инструктаж по правилам пользования компьютера.

2.2. Следует убедиться в отсутствии видимых повреждений аппаратуры и рабочего места. Визуально проверить исправность штепсельной розетки, целостность проводов питания, штепсельной вилки.

3. Требования безопасности во время работы.

3.1. Необходимо соблюдать оптимальное расстояние от глаз до экрана монитора (60-70 см.), допустимо не менее 50 см.

3.2. При работе на компьютере следует сидеть прямо, с небольшим наклоном вперед, не сутулясь. Величина угла в суставах не должна быть менее 90°.

3.3. При внезапном отключении электроэнергии в сети выключить компьютер.

3.4. Во время эксплуатации при повреждении штепсельного соединения, токопроводящего кабеля, появлении дыма (огня) из компьютера, обнару-жения замыкания на корпус следует немедленно отключить компьютер и доложить о поломке преподавателю или дежурному лаборанту.

3.5. Оптимальное время непрерывной работы на компьютере 40-50 минут. По истечении этого времени необходимо сделать перерыв на 10 минут.

3.6. Строго запрещается: трогать разъёмы соединительных кабелей, прикасаться к экрану и к тыльной стороне монитора, клавиатуры, включать и выключать аппаратуру без указания преподавателя,

3.9. Строго запрещается класть книги, тетради на монитор и клавиатуру, работать во влажной одежде и влажными руками.

4. Требования безопасности при аварийной обстановке.

4.1. Необходимо выключить компьютер.

4.2. Нельзя пользоваться компьютером до полного устранения неисправности.

4.3. При получении травмы и внезапном заболевании следует немедленно известить преподавателя или лаборанта.

5. Требования безопасности по окончании работы.

5.1. Необходимо выключить компьютер.

5.2. Обо всех замечаниях и недостатках в работе компьютера следует сообщить преподавателю или дежурному лаборанту.

Лабораторная работа № 1

Введение в программирование на PHP

Цель и задачи работы:

Изучить вводные сведения языка PHP.

Содержание работы:

Создание простой PHP программы.

Оборудование, технические и инструментальные средства:

ПК, Программы Notepad++ (Блокнот), браузер и программа xampp.

Краткие теоретические сведения

Определения и термины

Web-программирование – разработка любых программных продуктов, предназначенных для работы на сайтах World Wide Web. Строго говоря, даже разработка web-страниц на чистом HTML является web-программированием, ведь при просмотре страницы браузер фактически исполняет код HTML, фор-матируя текст согласно инструкциям этого языка. В настоящее время под web-программированием понимают создание CGI-приложений и использова-ние на web-странице технологий JavaScript и VBScript для достижений сложных эффектов.

Web-сервер – программа, запущенная на узле сети Интернет и выдающая посетителям этого узла web-страницы по запросам. Также web-сервером часто называется узел, на котором эта программа запущена, или даже ком-пьютер, являющийся таким узлом.

CGI - (сокращение от Common Gateway Interface) - технология, позволяю-щая запускать на web-сервере программы, имеющие возможность получать данные от посетителей сайтов, поддерживаемых этим web-сервером, и в свою очередь выдавать им обработанные данные в виде web-страниц или других файлов.

Для использования технологии CGI программа web-сервер должна удов-летворять определенным критериям – " поддерживать CGI". Если программа, запускаемая на web-сервере, представлена не двоичным кодом (т. е. скомпи-лированным файлом), а текстовым, то для ее выполнения требуется *програм-ма-интерпретатор* того языка, на котором написана эта программа.

Такой интерпретатор включается в состав web-сервера и вызывается им при необходимости заняться выполнением программного кода. CGI-сценарий (CGI-скрипт) – программа (в текстовом виде), предназначенная для исполне-ния на web-сервере. Для создания CGI-скриптов можно использовать любой язык программирования – важно лишь, чтобы на том web-сервере, где пред-полагается эту программу запускать, имелся интерпретатор этого языка.

Препроцессор – программа, работающая совместно с web-сервером, кото-рая просматривает все или некоторые файлы, выдаваемые web-сервером по-сетителям, и выполняет над ними определенные действия в зависимости от содержащихся в этих файлах инструкций. PHP является именно препроцес-сором, что, собственно, и видно из его названия.

Все языки программирования, используемые при разработке web-сайтов, можно разделить на две большие группы.

К *первой* относятся те из них, код которых выполняется на компьютере посетителя сайта, т. е. в браузере, запущенном на компьютере пользователя. Это известные всем JavaScript и VBScript. Программы на этих языках встраивают-ся в код web-страниц или выносятся в отдельный файл, обращение к которо-му осуществляется из web-страницы (в этом случае браузер все равно обраба-тывает такие «вынесенные» программы таким же образом, как если бы они были встроены в код страницы).

Во *вторую* группу включаются те языки, программы на которых выполняются на том компьютере, где расположен web-сервер. Эта группа более обширна - дело в том, что в принципе на web-сервере могут исполняться програм-мы на любом языке, даже командных .bat-файлов MS-DOS, важно лишь, что-бы на нем была установлена программа-интерпретатор этого языка, удовлет-воряющая стандарту CGI, которому также должен удовлетворять сам web-сервер.

PHP относится ко второй группе - программа на PHP исполняется на web-сервере. Однако от других CGI-языков PHP сильно отличается в лучшую сторону прежде всего своей простотой. Впрочем, если исходить из механизма действия, то PHP более правильно называть не "CGI-языком", а препроцессо-ром – что, собственно, отражено даже в его названии.

В то время как CGI-приложение просто выдает некие данные в браузер посетителя, препроцессор просматривает все или некоторые файлы, выдаваемые web-сервером посетителю, и ищет в них определенные команды, которые и выполняет. Именно такой способ работы и позволяет указывать код программ на PHP непосредственно в тексте web-страниц.) Одним из наиболее заметных достоинств PHP является возможность без особых затруднений работать с серверами баз данных.

Синтаксис PHP

Синтаксис PHP во многом заимствован из таких языков как C, Java и Perl. Файл, обрабатываемый сервером как правило имеет расширение php.

PHP-код включаются в html-код в следующем виде:

```
<? PHP  текст_кода  ?>
```

или

```
<? текст_кода; ?>
```

Комментарии

PHP поддерживает комментарии 'C', 'C++' и оболочки Unix. Например:

```
<? php echo "This is a test"; // Это однострочный комментарий в
стиле c++
/* Это многострочный комментарий,
это ещё одна его строка */
echo "This is yet another test"; echo "One Final Test";
# Это комментарий в shell-стиле
?>
```

echo

```
<? php echo "Эта информация будет выведена в HTML"; ?>
```

Присвоение значений переменным

Переменные в программах на PHP, отделяются символами $.

```
$city = "Kokshetau";
```

`city` – переменная

`Kokshetau` – значение

Некоторые операции

арифметические:

`$a + $b` Сложение Сумма `$a` и `$b`.

`$a - $b` Вычитание Разность `$a` и `$b`.

`$a * $b` Умножение Произведение `$a` и `$b`.

`$a / $b` Деление Частное от деления `$a` на `$b`.

`$a % $b` Modulus Целочисленный остаток от деления `$a` на `$b`.

строковые:

Имеются две строковые операции. Первая – операция (`'.'`), которая возвращает объединение из правого и левого аргументов. Вторая – операция присвоения (`'.='`), которая присоединяет правый аргумент в левому аргументу.

```
$a="Hello "; $b=$a."World!"; // теперь $b содержит "Hello
//World!"
```

```
$a = "Hello "; $a .= "World!"; // теперь $a содержит "Hello
//World!"
```

`echo ("текст")` – вывод на web-страницу какого-либо текста. Чтобы вывести на web-страницу значение какой-либо переменной, достаточно просто написать ее имя внутри выводимой строки: команда `echo "это цифра $a"` выведет в web-страницу текст `"это цифра 1"`, если ранее переменной `$a` было присвоено значение, равное единице. В случае необходимости использовать в выводимой строке кавычки или иные специальные символы перед этими символами следует ставить символ " .

`if (условие) {...команды, которые должны выполняться, если условие верно...;} else {...команды, которые должны выполняться, если условие неверно...}` – команда, позволяющая выполнить то или иное действие в зависимости от истинности верности или ложности того или иного условия. В фигурных скобках может располагаться несколько команд, разделенных точкой с запятой.

`for (начальное значение счетчика, условие продолжения цикла, изменение счетчика на каждом цикле) { ...команды... ;}` - цикл, т. е. повторение указанных в нем команд столько раз, сколько позволит условие изменения счетчика цикла (т. с. переменной, специально выделенной для подсчета числа выполнений команд цикла).

Программа на PHP может прерываться кодом web-страницы - для этого достаточно вставить закрывающий тэг до этого кода и открывающий - после. Все, что находится между ними, будет выдаваться в браузер без какой-либо обработки, рассматриваясь как выводимое с помощью команды `echo`. Иными словами, код

```
<?php if ($a==1) { ?><p>Переменная a равна 1</p><?php> }?>
```

эквивалентен коду

```
<?php if ($a==1) {echo "<p> Переменная a равна 1</p>";}?>
```

Однако, первый вариант меньше нагружает процессор компьютера, на котором расположен интерпретатор PHP. Из сказанного также следует, что все программы на PHP, расположенные на одной web-странице, представляют собой одну большую программу, несмотря на то, что они разделяются блоками обычного текста страницы.

Функции

`date('j F Y')` – показывает текущую дату;
`date('H:i:s')` – показывает текущее время;

Порядок выполнения работы:

Упражнение 1. Создание простой программы на PHP

Задание 1. Установить программный комплекс xampp.

1. Запустите на выполнение xampp. Выполнится установка xampp-а.
2. Создайте папку www в папке `c:\xampp\htdocs`
3. Наберите в Блокноте следующий тестовый текст:

```
<html>
<head>
<title>Моя первая Web-страница</title>
</head>
<body>
<h2>Мой первый HTML-документ</h2>
<!-- Обойдемся без Hello, World! -->
Hello, <i>World Wide Web!</i>
<p>Это – моя первая Web-страница</p>
</body>
```

```
</html>
```

4. Сохраните текст в файле `first.php` в папке `www`.
5. Запустите на выполнение браузер и в адресной строке браузера введите адрес страницы `localhost/www/first.php`.

Задания для самостоятельной работы:

1. Создайте php-скрипт, который создает страницу "`Hello World`!!!";
2. Получите информацию о настройках php с помощью команды `phpinfo();`
3. Создайте php-скрипт, который выводит системное время и дату;
4. Создайте отдельную страницу, на которой будут размещаться ссылки на php скрипты;
5. Создайте php-скрипт, который создает страницу с таблицей возможных цветов HTML;
6. Создайте php-скрипт, который формирует web-страницу с таблицей умножения;

Правила отчетности студента:

Выполнить все лабораторные работы и показать преподавателю результаты работ полученные во время выполнения.

Контрольные вопросы:

1. Как указываются комментарии в PHP?
2. Назовите функции даты и времени в PHP?
3. Какие арифметические операции используются в PHP?

Список использованных источников:

1. Коггзолл, Джон. PHP 5. Полное руководство.: Пер. англ. – М: Издательский дом «Вильямс», 2006. – 750 с.: ил. – Парал. тит. англ.
2. Котеров Д. В. Самоучитель PHP 4. – СПб: БХВ-Петербург, 2001. – 576 с.: ил.

Лабораторная работа № 2

Основы синтаксиса PHP

Цель и задачи работы:

Изучить основные элементы синтаксиса языка PHP.

Содержание работы:

Создание PHP программы с основными элементами синтаксиса.

Оборудование, технические и инструментальные средства:

ПК, Программы Notepad++ (Блокнот), браузер и программа xampp.

Краткие теоретические сведения

Типы данных в PHP

PHP поддерживает четыре скалярных (Integer, Double, Boolean, String) и два смешанных (Array, Object) типа данных.

Некоторые операции PHP

инкремента/декремента;

`++$a` Pre-increment Увеличивает `$a` на `1`,затем возвращает `$a`.
`$a++` Post-increment Возвращает `$a`, затем увеличивает `$a` на `1`.
`--$a` Pre-decrement Уменьшает `$a` на `1`, затем возвращает `$a`.
`$a--` Post-decrement Возвращает `$a`, затем уменьшает `$a` на `1`.

арифметические:

`$a + $b` Сложение Сумма `$a` и `$b`.
`$a - $b` Вычитание Разность `$a` и `$b`.
`$a * $b` Умножение Произведение `$a` и `$b`.
`$a / $b` Деление Частное от деления `$a` на `$b`.
`$a % $b` Modulus Целочисленный остаток от деления `$a` на `$b`.

строковые:

Имеются две строковые операции. Первая – *операция ('.')*, которая возвращает объединение из правого и левого аргументов. Вторая – *операция присвоения ('.=')*, которая присоединяет правый аргумент к левому аргументу.

```
$a="Hello "; $b=$a."World!"; // теперь $b содержит "Hello World!"
$a="Hello "; $a.="World!";   // теперь $a содержит "Hello World!"
```

Выражения сравнения

Выражения сравнения вычисляются в `0` или `1`, означая `FALSE` или `TRUE` (соответственно).

PHP поддерживает
> (больше),
>= (больше или равно),
== (равно),
!= (не равно),

< (меньше) и <= (меньше или равно).

Эти выражения чаще всего используются внутри условных операторов, таких как if.

сравнения:

`$a==$b` равно TRUE, если `$a` равно `$b`.

`$a!=$b` не равно TRUE, если `$a` не равно `$b`.

`$a<>$b` не равно TRUE, если `$a` не равно `$b`.

`$a<$b` меньше TRUE, если `$a` строго меньше `$b`.

`$a>$b` больше TRUE, если `$a` строго больше `$b`.

`$a<=$b` меньше или равно TRUE, если `$a` меньше или равно `$b`.

`$a>=$b` больше или равно TRUE, если `$a` больше или равно `$b`.

Функции

`gettype(переменная)` – определение типа данных;

Функции проверки на соответствие определенному типу данных:

`is_integer(), is_double(), is_string(), is_bool(),`
`is_array(), is_object().`

`is_numeric()` – проверка правильности записи числа в строке;

`settype(переменная,тип)` – устанавливает определенный тип;

Специальные функции для прямого и обратного преобразования чисел в десятичную систему из других наиболее востребованных систем с основа-ниями `2`, `8` и `16`:

`bindec()` и `decbin()`, `octdec()` и `decoct()`,`hexdec()` и `dechex()`;

Порядок выполнения работы:

Упражнение 1. Создание программы с переменными

Задание 1. Выполнить следующий листинг программы с профессиональной вставкой.

```
<?php
if ($expr)
{
echo "<b>вывод средствами PHP</b>";
}
else
{
?>
<b>Профессиональная   вставка</b>
<?php
}
?>
```

Смысл конструкции `if ... else` состоит в том, что в зависимости от выражения `$expr` будет выполняться то или иное действие. В данном случае действиями являются выполнение инструкции `echo` и альтернативный вывод текста, называемый профессиональной вставкой. Суть ее заключается в том, что после закрывающего тега (`?>`) строки передаются для обработки браузером до тех пор,

пока не встретится еще один открывающий тег (<?php). Профессиональная вставка в основном используется для вывода текста большого размера. Обратите внимание, что в нашем примере при значении $expr равном 1 будет выполняться только инструкция echo.

Задание 2. Выполнить программу, в которой показана возможность вставки HTML-кода в PHP-код.

```
<?php
echo "<b>Hello, World!</b>";
?>
```

Упражнение 2. Создание программы с различными типами данных

Задание 1. Выполнить программу с различными типами данных.

```
<?php
// в десятичной системе все три числа равны 317
$oct_int_num=0475; //восьмеричные числа начинаются с 0
echo $oct_int_num;
$dec_int_num=317;  //эта запись десятичного числа
echo $oct_int_num;
$hex_int_num=0x13D;//шестнадцатеричные числа начинаются с 0x
echo $oct_int_num;
?>
```

Задания для самостоятельной работы:

1. Создайте скрипт, в котором переменной $test_var попеременно присвой-те данные типов Integer, Double, Boolean, String, выводя очередной тип в браузер с определением очередного типа;
2. Создайте скрипт, в котором переменной $test_var попеременно присвой-те данные типов Integer, Double, Boolean, String, выводя очередной тип в браузер с проверкой на соответствие текущему типу переменной;
3. Создайте скрипт, в котором переменной $test_var присвойте данное типа Double (например 12,54) затем поочередно преобразуйте это данное в типы String, Integer, Boolean, выводя значение переменной очередного типа в браузер и выводя затем тип этой переменной;
4. Создайте скрипт, который преобразует в Boolean следующие значения:
   - целое число 0;
   - дробное число 0.0;
   - пустую строку или строку 0;
   - любые другие значения.
5. Создайте скрипт, который преобразует в Integer следующие значения:
   - значение FALSE и значение TRUE;
   - вещественное число с дробной частью;
   - строку не начинающуюся с цифры и строку начинающуюся с цифры.
6. Создайте скрипт, который преобразует в String следующие значения:
   - значение FALSE и значение TRUE;
   - любые числа преобразуйте в строку;
7. Необходимо преобразовать число из одной системы счисления в другую.

Правила отчетности студента:

Выполнить все лабораторные работы и показать преподавателю результаты работ полученные во время выполнения.

Контрольные вопросы:

1. Какая функция определяет тип переменной?
2. Какая функция проверяет тип переменной?
3. Какая функция устанавливает тип?
4. Каким образом будут преобразованы в `Boolean` целое `0`, дробное `0,0`, пустая строка или строка `0` и любые другие значения?
5. Каким образом будут преобразованы в `Integer` значение `FALSE` и `TRUE`, вещественное число с дробной частью, строка которая не начинается с цифры и наоборот?

Список использованных источников:

1. Коггзолл, Джон. PHP 5. Полное руководство.: Пер. англ. – М: Издательский дом «Вильямс», 2006. – 750 с.: ил. – Парал. тит. англ.
2. Котеров Д. В. Самоучитель PHP 4. – СПб: БХВ-Петербург, 2001. – 576 с.: ил.

Лабораторная работа № 3

Операторы PHP

Цель и задачи работы:

Изучить операторы языка PHP.

Содержание работы:

Создание PHP скриптов с операторами PHP.

Оборудование, технические и инструментальные средства:

ПК, Программы Notepad (Блокнот), браузер и программа xampp.

Краткие теоретические сведения

Определения и термины

Некоторые операции

инкремента/декремента;

`++$a` Pre-increment Увеличивает `$a` на `1`,затем возвращает `$a`.
`$a++` Post-increment Возвращает `$a`, затем увеличивает `$a` на `1`.
`--$a` Pre-decrement Уменьшает `$a` на `1`, затем возвращает `$a`.
`$a--` Post-decrement Возвращает `$a`, затем уменьшает `$a` на `1`.

арифметические:

`$a + $b` Сложение Сумма `$a` и `$b`.
`$a - $b` Вычитание Разность `$a` и `$b`.
`$a * $b` Умножение Произведение `$a` и `$b`.
`$a / $b` Деление Частное от деления `$a` на `$b`.
`$a % $b Modulus` Целочисленный остаток от деления `$a` на `$b`.

строковые:

Имеются две строковые операции. Первая - операция ('.'), которая возвращает объединение из правого и левого аргументов. Вторая - операция присвоения ('.='), которая присоединяет правый аргумент в левому аргументу.

```
$a = "Hello "; $b = $a."World!"; // теперь $b содержит "Hello
   World!"
$a = "Hello "; $a .= "World!";   // теперь $a содержит "Hello
   World!"
```

Выражения сравнения

Выражения сравнения вычисляются в 0 или 1, означая FALSE или TRUE (соответственно).

PHP поддерживает
> (больше),
>= (больше или равно),
== (равно),

!= (не равно),
< (меньше) и <= (меньше или равно).
Эти выражения чаще всего используются внутри условных операторов, таких как if.

сравнения:

$a == $b равно TRUE, если $a равно $b.
$a != $b не равно TRUE, если $a не равно $b.
$a <> $b не равно TRUE, если $a не равно $b.
$a < $b меньше TRUE, если $a строго меньше $b.
$a > $b больше TRUE, если $a строго больше $b.
$a <= $b меньше или равно TRUE, если $a меньше или равно $b.
$a >= $b больше или равно TRUE, если $a больше или равно $b.

Некоторые операторы

```
include "имя файла"
```

- команда для включения содержимого одного файла в другой. Содержимое файла, имя которого указывается в команде, целиком и полностью вставляется на то место, где располагается эта команда, при этом все коды PHP, содержащиеся во вставляемом файле, исполняются так же, как если бы они были на месте этой команды. (Помните, что файл именно вставляется – т. е., например, пути к картинкам, которые должны присутствовать во вставляемом файле, следует указывать от местонахождения того файла, в котором находилась команда include.) Если файл, включаемый в страницу при помощи команды include, отсутствует, то вместо него размещается уведомление об этом, а программа на PHP выполняется дальше. (При необходимости завершения обработки и выдачи web-страницы в случае отсутствия включаемого файла, вместо команды include следует использовать команду require.)

```
mail ("Кому", "Тема", "Текст сообщения", "Дополнительные заго-
ловки")
```

- отправка почтового сообщения. При выполнении данной команды на сервере в соответствии с указанными параметрами формируется электронное письмо и отправляется с помощью установленной на сервере почтовой программы. В качестве параметра "Кому" может выступать набор адресов, разделенных запятыми. "Дополнительные заголовки" могут быть любые (естественно, допустимые почтовыми протоколами!), разделяться они должны должны комбинацией символов /n, которая в PHP означает перевод строки. (Если среди "Дополнительных заголовков" не указано поле From, то оно заполняется по умолчанию почтовой программой web-сервера, например, именем "Unprivileged User".)

```
echo("текст")
```

- вывод на web-страницу какого-либо текста. Чтобы вывести на web-страницу значение какой-либо переменной, достаточно просто написать ее имя внутри выводимой строки: команда echo "это цифра $a" выведет в web-страницу текст

"это цифра 1", если ранее переменной $a было присвоено значение, равное единице. В случае необходимости использовать в выводимой строке кавычки или иные специальные символы перед этими символами следует ставить символ ".

Порядок выполнения работы:

Упражнение 1. Создание скрипта с операторами

Задание 1. Выполнить следующий листинг программы.

1. Наберите следующий листинг:

```
<?php
define (CONSTANT,"hello");
if (defined ("CONSTANT"))
{
echo ("Константа определена");
echo("<br>");
echo (CONSTANT);
}
echo("<br>");
$a=5;
$b=2;
$c="abc";
echo ("<br>");
echo(gettype($a));
echo("<br>");
echo (gettype($b));
echo("<br>");
echo (gettype($c));
echo("<br>");
settype($a,string);
$c=(integer)$c;
echo("<br>");
echo (gettype($a));
echo("<br>");
echo (gettype($c));
echo ("<br>");
echo("<br>");
echo($c);
echo("<br>");
echo($a%$b);
echo("<br>");
echo($b++);
echo("<br>");
echo($b);
echo("<br>");
echo(6&$a);
echo("<br>");
echo(++$a-1);
echo("<br>");
echo($a);
echo("<br>");
echo($a|=5);
```

```
   echo("<br>");
   echo($a+=1);
   echo("<br>");
   echo($b*($a%$b)+$b);
   echo("<br>");
   echo(10-(++$b)+($a%3)*2);
   echo("<br>");
   echo("<br>");
?>
</body>
</html>
```

2. В результате, когда вы запустите код на просмотр, у вас должно получиться следующее:

```
Константа определена
hello

integer
integer
string

string
integer

0
1
2
3
4
5
6
7
8
9
10
```

Задания для самостоятельной работы:

1. Скопируйте весь код листинга задания 1 на новую созданную страницу, сохраните ее под именем «2.php» и отредактируйте ее так, чтобы в результате, когда Вы запустите код на просмотр у Вас должны выводиться на экран только нечетные числа.

2. Создайте листинг «3.php». Снова скопируйте код листинга «1.php» на новый созданный листинг «3.php» и отредактируйте ее так, чтобы в результате, когда Вы запустите код на просмотр, у Вас должны выводиться на экран только четные числа.

Правила отчетности студента:

Выполнить все лабораторные работы и показать преподавателю результаты работ полученные во время выполнения.

Контрольные вопросы:

1. Перечислите операции инкремента и декремента.
2. Перечислите арифметические операции.
3. Перечислите строковые операции.
4. Какие выражения сравнения существуют в PHP?
5. Какие циклические команды имеются в PHP?

Список использованных источников:

1. Коггзолл, Джон. PHP 5. Полное руководство.: Пер. англ. – М: Издательский дом «Вильямс», 2006. – 750 с.: ил. – Парал. тит. англ.
2. Котеров Д. В. Самоучитель PHP 4. – СПб: БХВ-Петербург, 2001. – 576 с.: ил.

Лабораторная работа № 4

Управляющие операторы PHP (условные операторы)

Цель и задачи работы:

Изучить условные операторы языка PHP.

Содержание работы:

Создание PHP скриптов с условными операторами PHP.

Оборудование, технические и инструментальные средства:

ПК, Программы Notepad++ (Блокнот), браузер и программа xampp.

Краткие теоретические сведения

Определения и термины

`if (условие) {...команды, которые должны выполняться, если условие верно...;} else {...команды, которые должны выполняться, если условие неверно...}`

- команда, позволяющая выполнить то или иное действие в зависимости от истинности верности или ложности того или иного условия. В фигурных скобках может располагаться несколько команд, разделенных точкой с запятой.

Для расширения возможностей условного оператора `if` в PHP ввели конструкцию `elseif`. В общем случае она выглядит так:

```
if (выражение_1) действие;  // выполняется, если выражение_1 равно TRUE
elseif (выражение_2) действие;  // выполняется,
                                //если выражение_1 равно FALSE и
                                // выражение_2 равно TRUE
else действие;                  // выполняется, если
                                // выражение_1 равно FALSE и
                                // выражение_2 равно FALSE
```

Данная конструкция позволяет проверить альтернативные условия. В действительности она введена для того, чтобы избежать многократной вложенности операторов `if`.

switch (выражение) {case значение: ... команды...; break; case другое значение: ... команды...; break;}

- оператор выбора. При его работе содержимое, заключённое в фигурные скобки, просматривается сверху вниз. Как только будет найден оператор `case` со значением, совпадающим со значением выражения, PHP начнёт выполнять весь код, следующий за этим оператором `case` до последней фигурной скобки оператора `switch` или до первого оператора `break`, в зависимости от того, что появится раньше. (Обратите внимание, что если команду `break` не указать в конце кода, относящегося к одному варианту значения выражения в заголовке оператора switch, PHP будет выполнять код дальше - т. е. тот, который принадлежит уже следующему оператору `case`! Это - одно из отличий данного оператора от аналогичных в других языках программирования.) В конце оператора `switch` можно

указать оператор `default`. Код, стоящий после него, выполнится в том случае, если значение выражения в заголовке оператора не совпадет ни с одним из значений после операторов `case`.

Программа на PHP может прерываться кодом web-страницы - для этого достаточно вставить закрывающий тэг до этого кода и открывающий - после. Все, что находится между ними, будет выдаваться в браузер без какой-либо обработки, рассматриваясь как выводимое с помощью команды `echo`. Иными словами, код

```
<?php if ($a==l) { ?><p>Переменная а равна 1</p><?php> }?>
```

эквивалентен коду

```
<?php if ($a==l) {echo "<p> Переменная а равна 1</p>";}?>
```

Однако, первый вариант меньше нагружает процессор компьютера, на котором расположен интерпретатор PHP. Из сказанного также следует, что все программы на PHP, расположенные на одной web-странице, представляют собой одну большую программу, несмотря на то, что они разделяются блока-ми обычного текста страницы. Именно поэтому переменная, объявленная в расположенном в начале страницы коде, сохраняет свое значение не только до ее конца, но и во всех присоединяемых с помощью команды include файлах.

Порядок выполнения работы:

Упражнение 1. Создание скрипта с условными операторами

Задание 1. Выполнить следующий листинг программы.

```
<?php
Sa = 1;
$b = 0;
if ($a > $b) echo "a больше b";
else echo "a меньше,  либо равно b";
?>
```

Задание 2. Особенности работы оператора `if`. Выполните следующие 2 листинга.

```
<?php
$a = 1;
$b =  0;
if ($a > $b)
  echo "a больше b";
else
  echo "a меньше, либо равно b";
echo "Это будет вьведено";
?>
```

и

```
<?php
$a = 2;
$b = 0;
if ($a > $b)
if (($a - $b) = = 1) echo "разность между a и b равна  1";
else echo "a меньше,  либо равно b";
?>
```

Задание 3. Особенности работы оператора `elseif`. Выполните следующий листинг.

```
<?php
if ($a > $b)       // $a больше $b?
{
  echo "a больше b";   // если да. то выводим и остальное про-
пускаем
  }
elseif ($a = = $b)     // если нет. то $a равно $b?
{
  echo "a равно b";    // если да, то выводим и
                       // остальное пропускаем
  }
else
{
  echo "a меньше b";   // если нет выводим
  }
?>
```

Задание 4. Выполните следующий листинг с оператором `elseif`.

```
<?php
$book_name = "Самоучитель по PHP"; // входные данные
if ( $book_name == "Самоучитель по Perl" )
{
  echo "Автор: Петров";            // выходные данные
   }
elseif ( $book_name == "Самоучитель по ASP" )
{
  echo "Автор: Иванов";            // выходные данные
  }
elseif ( $book_name == "Самоучитель по PHP" )
{
  echo "Автор: Сидоров";           // выходные данные
  }
?>
```

Задание 5. Выполните следующий листинг с оператором `case`.

```
<?php
$book_name  =  "Самоучитель  no  PHP";
switch ($book_name)
  {
  // выводится,если $book_name имеет значение
  // "Самоучитель по Perl"
  case "Самоучитель по Perl":  echo "Автор:  Петров";
  // выход  из  конструкции  switch
  break;
  case "Самоучитель по ASP":  echo "Автор:  Иванов";
  break;
  case "Самоучитель no PHP": echo "Автор: Сидоров";
  break;
  // сообщение выведется, если не было совпадений
  default: echo "Такой книги в наличии нет";
  }
```

```
?>
```

Присутствие в конструкции `switch` слов `default` и `break` не является обязательным. Если опустить слово `default`, то в случае, когда совпадений не было, никаких действий просто не выполняется. Интереснее дело обстоит с оператором `break` этои случай рассматривается в следующем задании.

Задание 6. Выполните следующий листинг с оператором `case`.

```
<?php
$book_name = "Самоучитель по Perl";
switch ( $book_name )
   {
      case "Самоучитель по Perl": echo  "Автор: Петров";
      case "Самоучитель по ASP":  echo  "Автор:   Иванов";
      break;
      case "Самоучитель по PHP":  echo "Автор:  Сидоров";
      break;
      default: echo "Такой книги в наличии нет";
   }
?>
```

В результате выведутся фамилии *Петров* и *Иванов*. Другими словами, если совпали выражения и опущен оператор `break`, то выполнятся все действия вплоть до следующего оператора `break`. В нашем случае это привело к неправильной работе программы. Но есть случаи, когда отсутствие `break` помогает избежать многократного повторения в конструкции `switch`. Например, *Петров* написал не только самоучитель по `Perl`, но и еще несколько книг, тогда задачу можно решить так, как показано на следующем листинге.

Задание 7. Выполните следующий листинг с оператором `case` с и особенностями оператора `break`.

```
<?php
$book_name = "Информатика в школе";
switch ( $book_name )
   {
      case "Самоучитель no Perl";
      case "Информатика в школе".
      case "Программирование в Internet":  echo "Автор:   Петров";
      break;
      case "Самоучитель no ASP": echo "Автор: Иванов";
      break;
      case "Самоучитель no PHP": echo "Автор: Сидоров";
      break;
      default: echo "Такой книги в наличии нет";
   }
?>
```

В этом случае если пользователь ввел название одной из трех книг Петрова, выведется его фамилия, так как отсутствует оператор `break`. Стоит отметить, что таким образом задача решается намного эффективнее, чем многократное повторение операторов `break` и вывода фамилии *Петров*.

Нельзя не сказать еще несколько слов о выражении, следующим за `case`. В отличие от многих других языков программирования, оно может иметь любой скалярный тип данных. Например, в решениях наших задач мы использовали строковые выражения.

Задания для самостоятельной работы:

1. Наберите следующий листинг:

```
<h3>Преобразование чисел в слова</h3>
<?php
mt_srand(time()+(double)microtime()*1000000);
$i=mt_rand(0,9);
switch ($i)
{
   case 1: $j=" один";
   break;
   case 2: $j=" два";
   break;
   case 3: $j=" три";
   break;
   case 4: $j=" четыре";
   break;
   case 5: $j=" пять";
   break;
   case 6: $j=" шесть";
   break;
   case 7: $j=" семь";
   break;
   case 8: $j=" восемь";
   break;
   case 9: $j=" девять";
   break;
   case 0: $j=" ноль";
   break;}
   echo($i."=".$j);
?>
```

   В листинге строки:

```
(mt_srand(time()+(double)microtime()*1000000);
$i=mt_rand(0,9);
```

   позволяют присваивать переменной `$i` случайные значения из диапазона `0-9`)
2. Сохраните программу.
3. Исправьте код программы, чтобы на экран выводилась строка, в которой надпись указывала бы, является число четным или нечетным?
4. Переделаем программу так, чтобы она переводила числа от `1` до `99`;
5. Переделайте код таким образом, чтобы программа могла работать и с числами большими в диапазоне `100 - 1000`.

Правила отчетности студента:

Выполнить все лабораторные работы и показать преподавателю результаты работ полученные во время выполнения.

Контрольные вопросы:

1. Перечислите операторы условий.
2. Для каких целей используется оператор `if`?
3. Назовите отличия оператора `if` от `elseif`.
4. Для каких случаев применяют оператор switch?
5. Какую роль выполняет оператор `break`?
6. Назначение оператора `default`?

Список использованных источников:

1. Коггзолл, Джон. PHP 5. Полное руководство.: Пер. англ. – М: Издательский дом «Вильямс», 2006. – 750 с.: ил. – Парал. тит. англ.
2. Котеров Д. В. Самоучитель PHP 4. – СПб: БХВ-Петербург, 2001. – 576 с.: ил.

Лабораторная работа № 5

Работа со строками в PHP

Цель и задачи работы:

Изучить работу со строками в PHP.

Содержание работы:

Создание PHP скриптов с обработкой строк.

Оборудование, технические и инструментальные средства:

ПК, Программы Notepad++ (Блокнот), браузер и программа xampp.

Краткие теоретические сведения

Определения и термины

Анатомия строки

Строка – это набор символов, которые рассматриваются как один объект. В PHP строки заключаются в кавички. Поэтому для того, чтобы определить строку, нужно присвоить переменную строку, заключенную в одиночные или двойные кавычки:

```
$phrase = "Пусть всегда будет мир!";
$phrase = 'Пусть всегда будет мир!';
```

Форматированный вывод строк

Функция `printf()` выводит строку в определенном формате. Пример:

```
<?php
  $str="Число 8 в двоичном представлении: %b";
  // выводит: Число 8 в двоичном представлении: 1000
  printf($str,8);
?>
```

Первый аргумент `printf()` – строка для вывода. Ее формат определяется с помощью сочетания специальных символов. Символ `%` ставится первым, и буква `b`, определяющая двоичный формат вывода целого числа. Это число передается в качестве следующего входного параметра. Сначала оно преобразуется к целому числу, а затем выводится в двоичном представлении.

Сим-вол	Описание
`b`	Параметр преобразуется в целое и выводится в виде двоичного числа
`c`	Параметр преобразуется в целое и выводится в виде символа с соответствующим кодом ASCII
`d`	Параметр преобразуется в целое и выводится в виде десятичного числа со знаком
`u`	Параметр преобразуется в целое и выводится в виде десятичного числа без знака

f	Параметр преобразуется в вещественное число и выводится в виде двоичного числа
o	Параметр преобразуется в целое и выводится в виде восьмеричного числа
s	Параметр преобразуется в строку
x	Параметр преобразуется в целое и выводится в виде шестнадцатеричного числа (в нижнем регистре)
X	Параметр преобразуется в целое и выводится в виде шестнадцатеричного числа (в верхнем регистре)

В строке имеется возможность вставить несколько различных комбинаций специальных символов, при этом их количество должно совпадать с числом параметров функции `printf()`, кроме первого.

Ещё одной важной функцией работы с текстом является функция `substr($str,$start,$length)`. Её назначение – возвращать участок строки `$str`, начиная с позиции `$start` и длиной `$length`. Если `$length` не задана, то подразумевается подстрока от `$start` до конца строки `$str`.

`Strlen($str)` возвращает длину строки, количество символов строки `$str`.

Порядок выполнения работы:

Упражнение 1. Создание скрипта с обработкой строк.

Задание 1. Выполнить следующий скрипт вывода числа в двоичном и восьмеричном представлении.

```
<?php
$str="Двоичное и восьмеричное представление числа 12: %b и %o";
// выведет: Двоичное и восьмеричное представление числа 12: 1100
// и 14
printf($str,12,12);
?>
```

Задание 2. Вывод строки в поле определенной ширины.

```
<?php
$str="%10s";
printf($str,"Hello");
?>
```

Задание 3. Выравнивание по левому краю.

```
<?php
$str="%-10s";
printf($str,"Hello");
?>
```

Задание 4. Определение длины строки

```
<?php
$str="Hello, World";
$len=strlen($str);
```

```
// посимвольный вывод строки
for ($i=0;$i<=$len;$i++)
  {
  echo $str[$i];
  echo "<br>";
  }
?>
```

Задание 5. Выполнить следующий листинг программы.

```
<body bgcolor=FFFFCC>
<h1 align=center>
<font color=blue>
<tt>Печать всей таблицы символов</tt>
</font>
</h1>
<?php
for ($i=0,$x=0; $x<16; $x++)
  {for ($y=0;$y<16;$y++)
     {$chars[$x][$y]=array ($i,chr($i));
      $i++;
     }
  }
?>
<table border=1 cellpadding=1 cellspacing=0 align=center>
<? for($y=0;$y<16;$y++)
   {
    ?>
      <tr>
      <? for  ($x=0;$x<16;$x++)
         {
          ?>
             <td>
             <?=$chars[$x][$y][0]?>:<b><tt>
             <?=$chars[$x][$y][1]?></tt></b>
             </td>
           <?
         }
       ?>
       </tr>
     <?
   }
?>
</table>
</body>
```

Задания для самостоятельной работы:

1. Используя функции `ord` и `substr` напишите программу, которая будет выводить коды символов, составляющих ваши фамилию, имя, отчество.

Правила отчетности студента:

Выполнить все лабораторные работы и показать преподавателю результаты работ полученные во время выполнения.

Контрольные вопросы:

1. Какая функция используется для преобразования строк.
2. Перечислите основные форматы преобразования.
3. Какая функция позволяет возвратить подстроку строки.
4. Перечислите функции, связанные с работой с текстом.

Список использованных источников:

1. Коггзолл, Джон. PHP 5. Полное руководство.: Пер. англ. – М: Издательский дом «Вильямс», 2006. – 750 с.: ил. – Парал. тит. англ.
2. Котеров Д. В. Самоучитель PHP 4. – СПб: БХВ-Петербург, 2001. – 576 с.: ил.

Лабораторная работа № 6

Работа с формами в PHP

Цель и задачи работы:

1. Изучить механизм обработки форм в PHP.
2. Изучить основные функции и свойства объектов языка PHP.

Содержание работы:

Создание PHP скриптов с обработкой форм.

Оборудование, технические и инструментальные средства:

ПК, Программы Notepad++ (Блокнот), браузер и программа xampp.

Краткие теоретические сведения

Определения и термины

Основы клиент-серверных технологий

Если речь идет о *сервере,* невольно всплывает в памяти понятие *клиента.* Все потому, что эти два понятия неразрывно связаны. Объединяет их компьютерная архитектура клиент-сервер. Обычно, когда говорят «*сервер*», имеют в виду *сервер в архитектуре клиент-сервер*, а когда говорят «*клиент*» – имеют в виду *клиент в этой же архитектуре*.

Суть *клиент-серверной архитектуры* в том, чтобы разделить функции между двумя подсистемами: *клиентом*, который отправляет запрос на выполнение каких-либо действий, и *сервером*, который выполняет этот запрос. Взаимодействие между клиентом и сервером происходит посредством стандартных специальных протоколов, таких как TCP/IP и z39.50. На самом деле протоколов очень много, они различаются по уровням.

Сервер представляет собой набор программ, которые контролируют выполнение различных процессов. Соответственно, этот набор программ установлен на каком-то компьютере. Часто компьютер, на котором установлен сервер, и называют **сервером**. Основная функция компьютера-сервера – по запросу клиента запустить какой-либо определенный процесс и отправить клиенту результаты его работы.

Клиентом называют любой процесс, который пользуется услугами сервера. **Клиентом** может быть как пользователь, так и программа. Основная задача клиента – выполнение приложения и осуществление связи с сервером, когда этого требует приложение. То есть клиент должен предоставлять пользователю интерфейс для работы с приложением, реализовывать логику его работы и при необходимости отправлять задания серверу.

Взаимодействие между клиентом и сервером начинается по инициативе клиента. Клиент запрашивает вид обслуживания, устанавливает сеанс, получает нужные ему результаты и сообщает об окончании работы.

Услугами одного сервера чаще всего пользуется несколько клиентов одновременно. Поэтому каждый сервер должен иметь достаточно большую производительность и обеспечивать безопасность данных.

Логичнее всего устанавливать сервер на компьютере, входящем в какую-либо сеть, локальную или глобальную. Однако можно устанавливать сервер и на отдельно стоящий компьютер (тогда он будет являться одновременно и клиентом и сервером).

Существует множество типов серверов. Вот лишь некоторые из них.

Видеосервер - такой сервер специально приспособлен к обработке изображений, хранению видеоматериалов, видеоигр и т.п. В связи с этим компьютер, на котором установлен видеосервер, должен иметь высокую производительность и большую память.

Поисковый сервер предназначен для поиска информации в Internet.

Почтовый сервер предоставляет услуги в ответ на запросы, присланные по электронной почте.

Сервер WWW предназначен для работы в Internet.

Сервер баз данных выполняет обработку запросов к базам данных.

Сервер защиты данных предназначен для обеспечения безопасности данных (содержит, например, средства для идентификации паролей).

Сервер приложений предназначен для выполнения прикладных процессов. С одной стороны взаимодействует с клиентами, получая задания, а с другой – работает с базами данных, подбирая необходимые для обработки данные.

Сервер удаленного доступа обеспечивает коллективный удаленный доступ к данным.

Файловый сервер обеспечивает функционирование распределенных ресурсов, предоставляет услуги поиска, хранения, архивирования данных и возможность одновременного доступа к ним нескольких пользователей.

Обычно на компьютере-сервере работает сразу несколько программ-сер-веров. Одна занимается электронной почтой, другая распределением файлов, третья предоставляет web-страницы.

Из всех типов серверов нас будет интересовать *сервер WWW*. Часто его называют *web-сервером*, *http-сервером* или даже просто *сервером*.

Web-сервер

Во-первых, это хранилище информационных ресурсов.

Во-вторых, эти ресурсы хранятся и предоставляются пользователям в соответствии со стандартами Internet (такими, как протокол передачи данных HTTP).

Работа с документами web-сервера осуществляется при помощи *браузера* (например, IE, Opera или Mozilla), который отсылает серверу запросы, созданные в соответствии с протоколом HTTP. В процессе выполнения задания сервер может связываться с другими серверами.

Далее, говоря «**сервер**», мы будем подразумевать **web-сервер**.

В качестве примеров web-серверов можно привести сервер **Apache** группы Apache, Internet Information Server (IIS) компании Microsoft, **SunOne** фирмы Sun Microsystems, **WebLogic** фирмы BEA Systems, **IAS** (Inprise Application Server) фирмы Borland, **WebSphere** фирмы IBM, **OAS** (Oracle Application Server).

Протокол HTTP и способы передачи данных на сервер Internet построен по многоуровневому принципу, от физического уровня, связанного с физически-ми аспектами передачи двоичной информации, и до прикладного уровня, обеспечивающего интерфейс между пользователем и сетью.

HTTP

HTTP (*HyperText Transfer Protocol*, протокол передачи гипертекста) – это протокол прикладного уровня, разработанный для обмена гипертекстовой информацией в Internet.

HTTP предоставляет набор методов для указания целей запроса, отправляемого серверу. Эти методы основаны на дисциплине ссылок, где для указания ресурса, к которому должен быть применен данный метод, используется ***универсальный идентификатор ресурсов*** (*Universal Resource Identifier, URI*) в виде ***местонахождения ресурса*** (*Universal Resource Locator, URL*) или в виде его ***универсального имени*** (*Universal Resource Name, URN*).

Сообщения по сети при использовании протокола HTTP передаются в формате, схожем с форматом почтового сообщения Internet (RFC-822) или с форматом сообщений **MIME** *(Multipurpose Internet Mail Exchange)*.

HTTP используется для коммуникаций между различными пользовательскими программами и программами-шлюзами, предоставляющими доступ к существующим Internet-протоколам, таким как *SMTP* (протокол электронной почты), *NNTP* (протокол передачи новостей), *FTP* (протокол передачи файлов), *Gopher* и *WAIS*. HTTP разработан для того, чтобы позволять таким шлюзам через промежуточные программы-серверы (*proxy*) передавать данные без потерь.

Запрос клиента

Протокол реализует принцип запрос/ответ. Запрашивающая программа – клиент инициирует взаимодействие с отвечающей программой – сервером и посылает запрос, содержащий:

- метод доступа;
- адрес URL;
- версию протокола;
- сообщение с информацией о типе передаваемых данных, информацией о клиенте, пославшем запрос, и, возможно, с содержательной частью (телом) сообщения.

Ответ сервера содержит:

- строку состояния, в которую входит версия протокола и код возврата (успех или ошибка);
- сообщение, в которое входит информация сервера, метаинформация (т.е. информация о содержании сообщения) и тело сообщения.

В протоколе не указывается, кто должен открывать и закрывать соединение между клиентом и сервером. На практике соединение, как правило, открывает клиент, а сервер после отправки ответа инициирует его разрыв.

Форма запроса клиента

Клиент отсылает серверу запрос в одной из двух форм: в *полной* или *сокращенной*. Запрос в первой форме называется соответственно *полным запросом*, а во второй форме – *простым запросом*.

Простой запрос содержит метод доступа и адрес ресурса. Формально это можно записать так:

<Простой-Запрос> := <Метод> <символ пробел>
<Запрашиваемый-URL> <символ новой строки>

В качестве метода могут быть указаны GET, POST, HEAD, PUT, DELETE и другие.

В качестве запрашиваемого URL чаще всего используется URL-адрес ресурса.

Пример простого запроса:

```
GET http://www.kz/
```

Здесь `GET` – это метод доступа, т.е. метод, который должен быть применен к запрашиваемому ресурсу, а `http://www.kz/` – это URL-адрес запрашиваемого ресурса.

Полный запрос содержит строку состояния, несколько заголовков (заголовок запроса, общий заголовок или заголовок содержания) и, возможно, тело запроса. Формально общий вид полного запроса можно записать так:

<Полный запрос> := <Строка Состояния>
(<Общий заголовок>|<Заголовок запроса>|
<Заголовок содержания>)
<символ новой строки>
[<содержание запроса>]

Квадратные скобки здесь обозначают необязательные элементы заголовка, через вертикальную черту перечислены альтернативные варианты.

Элемент *<Строка состояния>* содержит метод запроса и URL ресурса (как и простой запрос) и, кроме того, используемую версию протокола HTTP. Например, для вызова внешней программы можно задействовать следующую строку состояния:

```
POST http://www.ru/cgi-bin/test HTTP/1.0
```

В данном случае используется метод POST и протокол HTTP версии 1.0.

В обеих формах запроса важное место занимает URL запрашиваемого ресурса. Чаще всего URL используется в виде URL-адреса ресурса. При обращении к серверу можно применять как полную форму URL, так и упрощенную.

В сокращенной форме опускают протокол и адрес сервера, указывая только местоположение ресурса от корня сервера. Полную форму используют, если возможна пересылка запроса другому серверу. Полная форма URL:

```
http://www.ru/cgi-bin/test?var1=value1&var2=value2
```

Если же работа происходит только с одним сервером, то чаще применяют сокращенную форму.

Методы

Как уже говорилось, любой запрос клиента к серверу должен начинаться с указания метода. Метод сообщает о цели запроса клиента. Протокол HTTP поддерживает достаточно много методов, но реально используются только три: POST, GET и HEAD.

Метод GET позволяет получить любые данные, идентифицированные с помощью URL в запросе ресурса. Если URL указывает на программу, то возвращается результат работы программы, а не ее текст (если, конечно, текст не есть результат ее работы). Дополнительная информация, необходимая для обработки запроса, встраивается в сам запрос (в строку статуса). При использовании метода GET в поле тела ресурса возвращается собственно затребованная информация (текст HTML-документа, например).

Существует разновидность метода GET – *условный GET*. Этот метод сообщает серверу о том, что на запрос нужно ответить, только если выполнено условие, содержащееся в поле `if-Modified-Since` заголовка запроса. Если говорить более точно, то тело ресурса передается в ответ на запрос, если этот ресурс изменялся после даты, указанной в `if-Modified-Since`.

Метод HEAD аналогичен методу GET, только не возвращает тело ресурса и не имеет условного аналога. Метод HEAD используют для получения информации о ресурсе. Это может пригодиться, например, при решении задачи тестирования гипертекстовых ссылок.

Метод POST разработан для передачи на сервер такой информации, как аннотации ресурсов, новостные и почтовые сообщения, данные для добавления в базу данных, т.е. для передачи информации большого объема и достаточно важной. В отличие от методов GET и HEAD, в POST передается тело ресурса, которое и является информацией, получаемой из полей форм или других источников ввода.

Использование HTML-форм для передачи данных на сервер

Для метода GET

При отправке данных формы с помощью метода GET содержимое формы добавляется к URL после знака вопроса в виде пар имя=значения, объединенных с помощью амперсанта &:

```
action?name1=value1&name2=value2&name3=value3
```

Здесь action – это URL-адрес программы, которая должна обрабатывать форму (это либо программа, заданная в атрибуте action тега form, либо сама текущая программа, если этот атрибут опущен). Имена name1,name2,name3 соответствуют именам элементов формы, а value1, value2, value3 – значениям этих элементов. Все специальные символы, включая = и &, в именах или значениях этих параметров будут опущены. Поэтому не стоит использовать в названиях или значениях элементов формы эти символы и символы кириллицы в идентификаторах.

Если в поле для ввода ввести какой-нибудь служебный символ, то он будет передан в его шестнадцатеричном коде, например, символ $ заменится на %24. Так же передаются и русские буквы.

Для полей ввода текста и пароля (это элементы input с атрибутом type=text и type=password), значением будет то, что введет пользователь. Если пользователь ничего не вводит в такое поле, то в строке запроса будет присутствовать элемент name=, где name соответствует имени этого элемента формы.

Для кнопок типа checkbox и radio button значение value определяется атрибутом VALUE в том случае, когда кнопка отмечена. Не отмеченные кнопки при составлении строки запроса игнорируются целиком. Несколько кнопок типа checkbox могут иметь один атрибут NAME (и различные VALUE), если это необходимо. Кнопки типа radio button предназначены для одного из всех предложенных вариантов и поэтому должны иметь одинаковый атрибут NAME и различные атрибуты VALUE.

В принципе создавать HTML-форму для передачи данных методом GET не обязательно. Можно просто добавить в строку URL нужные переменные и их значения.

```
http://www.ru/test.php?id=10&user=pit
```

В связи с этим у передачи данных методом GET есть один существенный недостаток – любой может подделать значения параметров. Поэтому не советуем использовать этот метод для доступа к защищенным паролем страницам, для передачи информации, влияющей на безопасность работы программы или сервера. Кроме того, не стоит применять метод GET для передачи информации, которую не разрешено изменять пользователю.

Несмотря на все эти недостатки, использовать метод GET достаточно удобно при отладке скриптов (тогда можно видеть значения и имена передаваемых переменных) и для передачи параметров, не влияющих на безопасность.

Для метода POST

Содержимое формы кодируется точно так же, как для метода GET, но вместо добавления строки к URL содержимое запроса посылается блоком данных как часть операции POST. Если присутствует атрибут ACTION, то значение URL, которое там находится, определяет, куда посылать этот блок данных. Этот метод, как уже отмечалось, рекомендуется для передачи больших по объему блоков данных.

Информация, введенная пользователем и отправленная серверу с помощью метода POST, подается на стандартный ввод программе, указанной в атрибуте action, или текущему скрипту, если этот атрибут опущен. Длина посылаемого файла передается в переменной окружения `CONTENT_LENGTH`, а тип данных – в переменной `CONTENT_TYPE`.

Передать данные методом POST можно только с помощью HTML-формы, поскольку данные передаются в теле запроса, а не в заголовке, как в GET. Соответственно и изменить значение параметров можно, только изменив значение, введенное в форму. При использовании POST пользователь не видит передаваемые серверу данные.

Основное преимущество POST запросов – это их большая безопасность и функциональность по сравнению с GET-запросами. Поэтому метод POST чаще используют для передачи важной информации, а также информации большого объема. Тем не менее не стоит целиком полагаться на безопасность этого механизма, поскольку данные POST запроса также можно подделать, например создав html-файл на своей машине и заполнив его нужными данными. Кроме того, не все клиенты могут применять метод POST, что ограничивает варианты его использования.

Переменные окружения

При отправке данных на сервер любым методом передаются не только сами данные, введенные пользователем, но и ряд переменных, называемых *переменными окружения*, характеризующих клиента, историю его работы, пути к файлам и т.п. Вот некоторые из переменных окружения:

- `REMOTE_ADDR` – IP-адрес хоста (компьютера), отправляющего запрос;
- `REMOTE_HOST` – имя хоста, с которого отправлен запрос;
- `HTTP_REFERER` – адрес страницы, ссылающейся на текущий скрипт;
- `REQUEST_METHOD` – метод, который был использован при отправке запроса;
- `QUERY_STRING` – информация, находящаяся в URL после знака вопроса;
- `SCRIPT_NAME` – виртуальный путь к программе, которая должна выполняться;
- `HTTP_USER_AGENT` – информация о браузере, который использует клиент.

Обработка запросов с помощью PHP

Внутри PHP-скрипта существует несколько способов получения доступа к данным, переданным клиентом по протоколу HTTP. До версии PHP 4.1.0 доступ

к таким данным осуществлялся по именам переданных переменных. Таким образом, если, например, было передано `first_name=Nina`, то внутри скрипта появлялась переменная `$first_name` со значением `Nina`. Если требовалось различать, каким методом были переданы данные, то использовались ассоциативные массивы `\$HTTP_POST_VARS` и `\$HTTP_GET_VARS`, ключами которых являлись имена переданных переменных, а значениями – соответственно значения этих переменных. Таким образом, если пара `first_name=Nina` передана методом GET, то

```
$HTTP_GET_VARS["first_name"]="Nina"".
```

Использовать в программе имена переданных переменных напрямую небезопасно. Поэтому было решено начиная с PHP 4.1.0 задействовать для обращения к переменным, переданным с помощью HTTP-запросов, специальный массив – `$_REQUEST`. Этот массив содержит данные, переданные методами POST и GET, а также с помощью *HTTP cookies*. Это *суперглобальный ассоциативный массив*, т.е. его значения можно получить в любом месте программы, используя в качестве ключа имя соответствующей переменной (элемента формы).

Пример

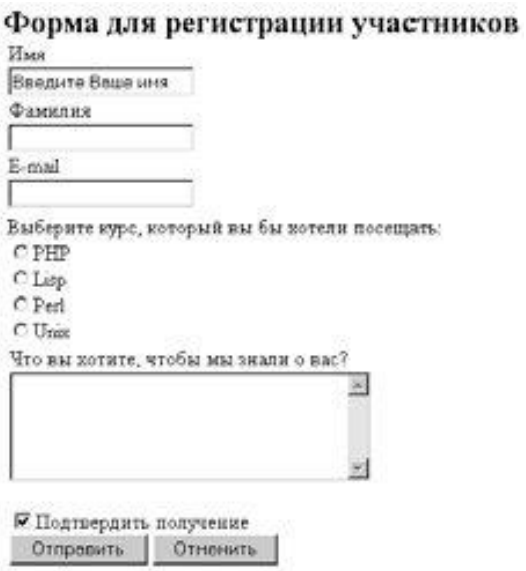

Рис. 2. Пример внешнего вида формы.

Создана форма для регистрации участников заочной школы программирования. Тогда в файле `action.php`, обрабатывающем эту форму, можно написать следующее:

```
<?php $str = "Здравствуйте,
    ".$_REQUEST["first_name"]. "
    ".$_REQUEST["last_name"]."! <br>";
$str .="Вы выбрали для изучения курс по
    ".$_REQUEST["kurs"];
echo $str; ?>
```

Тогда, если в форму мы ввели имя «Вася», фамилию «Петров» и выбрали среди всех курсов курс по PHP, на экране браузера получим такое сообщение:

Здравствуйте, Вася Петров!

Вы выбрали для изучения курс по PHP

После введения массива `$_REQUEST` массивы `$HTTP_POST_VARS` и `$HTTP_GET_VARS` для однородности были переименованы в `$_POST` и `$_GET` соответственно, но сами они из обихода не исчезли из соображений совместимости с предыдущими версиями PHP. В отличие от своих предшественников, массивы `$_POST` и `$_GET` стали суперглобальными, т.е. доступными напрямую и внутри функций и методов.

Приведем пример использования этих массивов. Допустим, нам нужно обработать форму, содержащую элементы ввода с именами `first_name`, `last_name`, `kurs` (например, форму, приведенную выше). Данные были переданы методом POST, и данные, переданные другими методами, мы обрабатывать не хотим. Это можно сделать следующим образом:

```
<?php $str = "Здравствуйте,
    ".$_POST ["first_name"]."
    ".$_POST ["last_name"] ."! <br>";
$str .= "Вы выбрали для изучения курс по ".
        $_POST["kurs"];
echo $str; ?>
```

Тогда на экране браузера, если мы ввели имя «Вася», фамилию «Петров» и выбрали среди всех курсов курс по PHP, увидим сообщение, как в предыдущем примере:

Здравствуйте, Вася Петров!

Вы выбрали для изучения курс по PHP

Для того чтобы сохранить возможность обработки скриптов более ранних версий, была введена директива `register_globals`, разрешающая или запрещающая доступ к переменным непосредственно по их именам. Если в файле настроек PHP параметр `register_globals=On`, то к переменным, переданным серверу методами GET и POST, можно обращаться просто по их именам (т.е. можно писать `$first_name`). Если же `register_globals=Off`, то нужно писать

```
$_REQUEST["first_name"]
```

или

```
$_POST["first_name"],
$_GET["first_name"],
$HTTP_POST_VARS["first_name"],
$HTTP_GET_VARS["first_name"].
```

С точки зрения безопасности эту директиву лучше отключать (т.е. `register_globals=Off`). При включенной директиве `register_globals` перечисленные выше массивы также будут содержать данные, переданные клиентом.

Иногда возникает необходимость узнать значение какой-либо переменной окружения, например метод, использовавшийся при передаче запроса или IP-адрес компьютера, отправившего запрос. Получить такую информацию можно с

помощью функции `getenv()`. Она возвращает значение переменной окружения, имя которой передано ей в качестве параметра.

```
<? getenv("REQUEST_METHOD");
    // возвратит использованный метод
echo getenv ("REMOTE_ADDR");
    // выведет IP-адрес пользователя,
    // пославшего запрос
?>
```

Как мы уже говорили, если используется метод GET, то данные передаются добавлением строки запроса в виде пар «имя_переменной=значение к URL-адресу ресурса». Все, что записано в URL после знака вопроса, можно получить с помощью команды

```
getenv("QUERY_STRING");
```

Благодаря этому можно по методу GET передавать данные в каком-нибудь другом виде. Например, указывать только значения нескольких параметров через знак плюс, а в скрипте разбирать строку запроса на части или можно передавать значение всего одного параметра. В этом случае в массиве `$_GET` появится пустой элемент с ключом, равным этому значению (всей строке запроса), причем символ «+», встретившийся в строке запроса, будет заменен на подчеркивание «_».

Методом POST данные передаются только с помощью форм, и пользователь (клиент) не видит, какие именно данные отправляются серверу. Чтобы их увидеть, хакер должен подменить нашу форму своей. Тогда сервер отправит результаты обработки неправильной формы не туда, куда нужно. Чтобы этого избежать, можно проверять адрес страницы, с которой были посланы данные. Это можно сделать опять же с помощью функции `getenv()`:

```
getenv("HTTP_REFERER");
```

Пример обработки запроса с помощью PHP

Нужно написать обработчики формы (см. выше) для регистрации участников заочной школы программирования и после регистрации отправить участнику сообщение.

```
<h2>Форма для регистрации студентов</h2>
<form action="1.php" method=POST>
Имя <br><input type=text name="first_name"
    value="Введите Ваше имя"><br>
Фамилия <br><input type=text name="last_name"><br>
E-mail <br><input type=text name="email"><br>
<p>Выберите курс, который вы бы хотели посещать:<br>
<input type=checkbox name='kurs[]' value='PHP'>PHP<br>
<input type=checkbox name='kurs[]' value='Lisp'>Lisp<br>
<input type=checkbox name='kurs[]' value='Perl'>PHP<br>
<input type=checkbox name='kurs[]' value='Unix'>Lisp<br>
<p>Что вы хотите, чтобы мы знали о вас? <br>
<textarea name="comment" cols=32 rows=5></textarea>
<input type=submit value="Отправить">
<input type=reset value="Отменить">
```

```
</form>
```

Следует отметить, способ передачи значений элемента `checkbox`. Когда мы пишем в имени элемента `kurs[]`, это значит, что первый отмеченный элемент `checkbox` будет записан в первый элемент массива `kurs`, второй отмеченный `checkbox` – во второй элемент массива и т.д. Можно, конечно, просто дать разные имена элементам `checkbox`, но это усложнит обработку данных, если курсов будет много.

Скрипт, который все это будет разбирать и обрабатывать, называется `1.php` (форма ссылается именно на этот файл, что записано в ее атрибуте action). По умолчанию используется для передачи метод GET, но мы указали POST. По полученным сведениям от зарегистрировавшегося человека, скрипт генерирует соответствующее сообщение. Если человек выбрал какие-то курсы, то ему выводится сообщение о времени их проведения и о лекторах, которые их читают. Если человек ничего не выбрал, то выводится сообщение о следующем собрании заочной школы программистов (ЗШП).

Порядок выполнения работы:

Упражнение 1. Создание программ обработки форм.

Задание 1. Реализуйте предложенный пример.
Задание 2. Создайте форму ввода данных о пользователе (ФИО, e-mail, телефон). Напишите скрипт, который проверяет правильность заполнения полей формы.

Задания для самостоятельной работы:

В ходе работы мы создадим калькулятор на языке PHP. Для этого будет написан скрипт, который будет выводить на экран форму для ввода данных и для выбора арифметического действия. Соответствующие вычисления будут производиться тем же скриптом после нажатия кнопки "Выполнить". В скрипте будут определены необходимые функции, каждая из которых будет вызываться для отработки соответствующего ей арифметического действия.

1. Выполнить следующий скрипт простейшего калькулятора:
   1.1. Откройте Блокнот, сохраните текущую страницу как calc.php
   Определим в теле скрипта функцию show(), которая будет выводить на экран форму для ввода данных.

```
<html>
<head>
<title>Калькулятор</title>
</head>
<body>
<?  /* приступаем непосредственно к php коду */
function show()
{
  global $action;
  ?>
  <FORM method="GET" action="calc.php">
  Первое число
  <input type="text" name="first">
```

```
  Второе число (Степень, процент)
  <input type="text" name="second">
  <select size="1" name="action">
    <option value="sum">Сложить</option>
    <option value="min">Вычесть</option>
    <option value="mult">Умножить</option>
    <option value="dev">Разделить</option>
    <option value="stepen">Возвести в степень</option>
    <option value="procent">Процент от числа</option>
    <option value="koren">Корень</option>
  </select>
  <br><input type="submit" value="Выполнить">
  </form>
  <?php
} //конец функции show()
```

Определим функцию `calc()`, которая будет производить арифметические действия в зависимости от выбора пользователя.

```
function calc()
{
global $action, $result, $first, $second;
switch($action)
 {
   case "sum": $result = $first+$second; break;
   case "min": $result = $first-$second; break;
   case "mult": $result = $first*$second; break;
   case "dev":
   if (!$second)  /* если второе число равно "0"
                     или вообще не введено */
  {
     exit("Извините, программа не может выполнить действие:
          на ноль делить нельзя");
   }
   $result=$first/$second; break;
   case "procent": $result = $first*($second/100); break;
   case "stepen": $result = pow($first, $second); break;
   case "koren": $result = pow($first,0.5); break;
 }
?>  //заканчиваем оператор switch
 Результат Вашего действия равен
<b> <? echo $result; ?>  </b> //вывод результата
 </font>
 <?
}  //конец функций calc()
Заканчиваем файл:
if ($action) calc(); else show();
?>
```

Последняя строка означает, что в том случае, если определена переменная `$action`, скрипт производит вычисления — иначе выводит на экран форму.

1.4. Разработать функции `exit()` и `pow()`.

2. Для придания калькулятору дополнительной функциональности:
   Поместите элементы формы в ячейки таблицы, сделайте так, чтобы порядок элементов не зависел от размеров окна браузера.
   Усовершенствуйте созданный калькулятор. Добавьте возможность вычислять одну из тригонометрических функций:
   - `cos`
   - `sin`
   - `acos`
   - `asin`
   - `tan`
   - `atan`

Правила отчетности студента:

Выполнить все лабораторные работы и показать преподавателю результаты работ полученные во время выполнения.

Контрольные вопросы:

1. Какие используются способы получения доступа к данным, переданным клиентом по протоколу HTTP?
2. Какие методы передачи данных используются между клиентом и сервером?

Список использованных источников:

1. Коггзолл, Джон. PHP 5. Полное руководство.: Пер. англ. – М: Издательский дом «Вильямс», 2006. – 750 с.: ил. – Парал. тит. англ.
2. Котеров Д. В. Самоучитель PHP 4. – СПб: БХВ-Петербург, 2001. – 576 с.: ил.

Лабораторная работа № 7

Обработка форм с помощью PHP

Цель и задачи работы:

1. Изучить механизм обработки форм в PHP.
2. Изучить механизм отправки сообщения с помощью почтового сервиса.

Содержание работы:

Создание PHP скриптов с обработкой форм и отправкой сообщений с помощью почтового сервиса.

Оборудование, технические и инструментальные средства:

ПК, Программы Notepad++ (Блокнот), браузер и программа xampp.

Краткие теоретические сведения

Доступ к значениям формы

Доступ к значениям формы в PHP можно получить с помощью специаль-ных (суперглобальных) массивов. Массивы `$_GET` и `$_POST` содержат значе-ния, отправленные, соответственно, методами `GET` и `POST`. Гибридный мас-сив `$_REQUEST` содержит значения из обоих массивов, а также значения массива `$_COOKIE`.

Элементу формы с определенным именем соответствует элемент массива `$_GET` или `$_POST` с соответствующим ключом. А значение этого элемента соответствует введенному пользователем значению для этого элемента.

Например, почтовый адрес из странички `comments.html` содержится в элементе `$_POST["email"]`, а текст комментария – в `$_POST["comments"]`.

Для типов `CHECKBOX` и `RADIO` атрибут `VALUE` определяет значение, которое получит PHP. Если установить флажок `may_contact`, значение `Y` будет содержать `$_POST["may_contact"]`. Если же этот элемент оставить не установ-ленным, он вообще не появится в соответствующем массиве. Поэтому нужно использовать функцию `isset`, чтобы узнать о том, установлен ли флажок.

Группа переключателей `gender` создает элемент `$_POST["gender"]`, который содержит значение `m` или `f`, в зависимости от выбранного значения. Если ни один из переключателей не отмечен, как в случае с флажком, соответствующий элемент не создается вообще.

С помощью функции `print_r` легко увидеть все значения, полученные из формы. Для этого на ее вход подается массив `$_POST`:

```
echo "<PRE>";
print_r($_POST);
echo "</PRE>";
```

Это очень полезный прием отладки, который позволяет увидеть все данные, которые сценарий получает из формы. Если создать файл send_comments.php и

поместить в него код, приведенный выше, на экране появятся значения каждого элемента формы. Вот пример вывода:

```
Array
(
  [name] => Петя Иванов
  [email] => peter@mail.net
  [gender] => m
  [refferer] => search
  [may_contact] => Y
  [comments] => Это мой любимый сайт
)
```

Даже значение кнопки подачи формы можно увидеть в PHP. Для этого нужно задать имя и отправить форму, щелкнув на нужной кнопке. Следующая форма содержит две кнопки с разными именами, а PHP позволяет определить, какая из них отправила форму:

```
<FORM ACTION="button.php" METOD=POST>
<INPUT TYPE="SUBMIT" NAME="button1" VALUE="Кнопка1">
<INPUT TYPE="SUBMIT" NAME="button2" VALUE="Кнопка2">
</FORM>
```

В `button.php` можно использовать следующую проверку для того, чтобы определить, на какой кнопке сделан щелчок:

```
if (isset($_POST["button1"]))
   {
     echo "Вы щелкнули на кнопке1";
   }
elseif (isset($_POST["button2"]))
   {
     echo "Вы щелкнули на кнопке2";
   }
else
   {
     echo "Я не знаю на какой кнопке вы щелкнули!";
   }
```

Атрибут `VALUE` кнопки подачи формы задает текст на кнопке, а также передается в PHP, когда щелкают на этой кнопке.

Элемент скрытого ввода

Есть еще один тип дескриптора `<INPUT>`, который используется для передачи данных между сценариями и при этом не отображается на Web-странице. Тип `HIDDEN` имеет имеет атрибуты NAME и VALUE, которые заменяют это значение. Он используется, чтобы подменить эти значения.

Следующий элемент скрытого ввода передается в PHP-сценарий при отправке формы, и элемент `$_POST["secret"]` содержит это значение:

```
<INPUT TYPE="HIDDEN" NAME="secret" VALUE="Это секрет">
```

Тип `HIDDEN` не подходит для передачи секретных паролей или другой важной информации. Не смотря на то что он не отображает на Web-странице, просмотр HTML-кода позволяет узнать эти значения.

Сценарий для отправки электронной почты

Функция mail

Функция отправляет сообщения с помощью почтового сервиса. Для корректной работы нужно задать имя почтового сервера в файле `php.ini`.

Для функции `mail` нужно задать три аргумента: *почтовый адрес получателя*, *тему* и *текст сообщения*. Четвертый необязательный аргумент задает дополнительные заголовки письма. Это позволяет указать специальные параметры `From:` или `Cc:`.

Сценарий `send_comments.php` задания 1 принимает данные из формы комментариев и отправляет их владельцу Web-сайта на почтовый ящик.

Сценарий проходит по всем значениям массива `$_POST` и создает строку `$body` для текста письма. Символ `\n` используется для разделения строчек, потому что письмо передается обычным текстом, без HTML-форматирования.

Письмо, которое получит владелец сайта, будет выглядеть примерно так:

```
Этот комментарий отправлен с помощью Web-сайта
name = Петя Иванов
email= peter@mail.net
gender = m
referrer = search
may_contact = Y
comments = Это мой любимый сайт
```

Порядок выполнения работы:

Упражнение 1. Создание программ обработки формы и отправки сообще-ния с помощью почтового сервиса.

Задание 1. Создать Web-форму для отправки пользовательских коммента-риев.

```
<FORM ACTION=”send_comments.php” METOD=POST>
<TABLE>
<TR>
   <TD>Ваше имя:</TD>
   <TD><INPUT TYPE=”TEXT” NAME=”name” SIZE=30></TD>
</TR>
<TR>
   <TD>Ваш почтовый адрес:</TD>
   <TD><INPUT TYPE=”TEXT” NAME=”email” SIZE=30></TD>
</TR>
<TR>
<TD>Ваш пол:</TD>
   <TD><INPUT TYPE=”RADIO” NAME=”gender” VALUE=”m”> Мужчина
       <INPUT TYPE=”RADIO” NAME=”gender” VALUE=”f”> Женщина
   </TD>
</TR>
<TR>
   <TD>
      <SELECT NAME=”referrer”>
      <OPTION VALUE=”search”>Поисковый сервер</OPTION>
```

```
      <OPTION VALUE="tv">Телевизионная реклама</OPTION>
      <OPTION VALUE="billboard">Доска объявлений</OPTION>
      <OPTION SELECTED VALUE="other">Другое</OPTION>
      </SELECT>
   </TD>
</TR>
<TR>
  <TD>Можно вам написать?</TD>
   <TD><INPUT TYPE="CHECKBOX" NAME="may_contact"
        VALUE="Y" CHECKED> </TD>
</TR>
<TR>
   <TD>Комментарии</TD>
   <TD><TEXTAREA ROWS=4 COLS=50 NAME="comments">Введите ваш
        комментарий здесь
       </TEXTAREA>
   </TD>
</TR>
</TABLE>
<INPUT TYPE="SUBMIT" VALUE="Отправить комментарий">
</FORM>
```

Задание 2. Создать сценарий `send_comments.php`, который принимает дан-ные из формы комментариев и отправляет их владельцу Web-сайта на почтовый ящик.

```
<?php
$body="Этот комментарий отправлен с помощью Web-сайта\n\n";
foreach($_POST as $field => $value)
  {
  $body .= sprintf("%s = %s\n", $field, $value);
   }
mail("owner@website.net", "Комментарий отправленный с помощью
Web-сайта", $body, 'From: "Web-комментарий"
<comments@website.net>');
?>
<H1>Спасибо</H1>
Ваш комментарий отправлен!
```

Задание 3. Реализуйте полностью вышеописанный пример.

Задания для самостоятельной работы:

1. Создайте форму ввода данных о пользователе сайта (ФИО, e-mail, телефон). Напишите скрипт, который отправляет письмо владельцу Web-сайта на почтовый ящик.

Правила отчетности студента:

Выполнить все лабораторные работы и показать преподавателю результаты работ полученные во время выполнения.

Контрольные вопросы:

1. Как можно получить доступ к значениям формы в PHP?

2. Какая функция используется для отправки отправки электронной почты?

Список использованных источников:

1. Веллинг, Люк, Томсон, Лора., Разработка Web-приложений с помощью PHP и MySQL, 3-е издание: Пер. с англ.– М: Издательский дом «Вильямс», 2006. – 880 с.: ил. – Парал. тит. англ.
2. Коггзолл, Джон. PHP 5. Полное руководство.: Пер. англ. – М: Издательский дом «Вильямс», 2006. – 750 с.: ил. – Парал. тит. англ.
3. Котеров Д. В. Самоучитель PHP 4. – СПб: БХВ-Петербург, 2001. – 576 с.: ил.
4. Харрис Э. PHP/MySQL для начинающих./Пер. англ. – М: Кудиц-Образ, 2005. – 384 с.

Лабораторная работа № 8

Динамическая генерация HTML- кода для форм

Цель и задачи работы:

1. Изучить механизм создания элементов форм с помощью PHP.
2. Изучить приемы позволяющие задавать значения по умолчанию для элементов ввода, а также генерировать раскрывающиеся списки и группы переключателей на основе данных сценария.

Содержание работы:

Создание PHP-скриптов динамически генерирующих элементы форм PHP.

Оборудование, технические и инструментальные средства:

ПК, Программы Notepad ++ (Блокнот), браузер и программа xampp.

Краткие теоретические сведения

HTML-формы и его элементы можно создавать с помощью PHP. Эти приемы позволяют задавать значения по умолчанию для элементов ввода, а также генерировать раскрывающиеся списки и группы переключателей на основе данных сценария.

Установка значений по умолчанию

Стандартные значения для ввода

Значения по умолчанию для текстового поля ввода задается в атрибуте VALUE. Это значение будет отображаться в форме после загрузки страни-цы, и если его не изменить, оно передается в PHP-сценарий при отправке формы.

Рассмотрим страницу электронного магазина с корзиной для покупок. У покупателя есть возможность изменить количество единиц товаров, перед окончательной отправкой заказа. Текущее количество товаров в корзине можно изменить в небольшом текстовом поле. После этого пользователь может щелкнуть на кнопке подачи формы и обновить содержимое корзины. Он содержит всего один товар и дает возможность выбрать его количество для покупки.

Установка атрибута CHECKED для типа CHECKBOX

С помощью PHP можно выполнить проверку для дескриптора `<INPUT>`, чтобы определить состояние атрибута `CHECKED` для типа `CHECKBOX`:

```
<INPUT TYPE=”CHECKBOX”
NAME=”mybox” <?php if(условие) echo “CHECKED”;?>>
```

Положение атрибута `CHECKED` не имеет значения, его можно поставить в другом месте для повышения удобочитаемости:

```
<INPUT <?php if(условие) echo “CHECKED”;?>
TYPE=”CHECKBOX” NAME=”mybox” >
```

При вставке PHP в HTML-код нужно внимательно следить за пробелами. Если в примере выше удалить пробел после закрытия дескриптора PHP и условие будет истинным, на выходе получим следующий HTML-код:

```
<INPUT CHECKEDTYPE=”CHECKBOX” NAME=”mybox”>
```

Название `CHECKEDTYPE` не является частью дескриптора `<INPUT>`, поэтому браузер вместо `CHECKBOX` выведет стандартный тип `TEXT`! Поэтому лучше всегда оставлять место вокруг динамических элементов в HTML-коде.

Установка переключателя

Атрибут `CHECKED` также используется для группы `RADIO`, чтобы установить элемент по умолчанию. Например, в электронном магазине можно предложить три варианта с различной ценой. Для того чтобы узнать, на каком варианте остановился покупатель, достаточно установить один вариант по умолчанию. При желании покупатель всегда сможет выбрать другой вариант (переключатель сбрасывается только при выборе другого из этой же группы):

```
<INPUT TYPE=”RADIO” CHECKED
NAME=”shipping” VALUE=”economy”> Экономный
<INPUT TYPE=”RADIO” NAME=”shipping” VALUE=”standard”>
Стандартный
<INPUT TYPE=”RADIO” NAME=”shipping” VALUE=”express”> Быстрый
```

Чтобы динамически установить атрибут `CHECKED` для одного из группы переключателей, нужно создать условие, которое проверит текущее значение переменной `$shipping` на совпадение с соответствующим элементом.

Установка стандартного значения для раскрывающегося списка

Атрибут `SELECTED` устанавливает дескриптор `<OPTION>` как элемент по умолчанию. Если в группе `<OPTION>` нет элемента с атрибутом `SELECTED`, первый устанавливается по умолчанию.

Можно использовать PHP для установки атрибута `SELECTED`, пройдя по всему списку дескрипторов `<OPTION>`.

Но результат будет таким же громоздким, как и для группы `RADIO`.

Создание элементов формы

С помощью специальных PHP-функций можно динамически генерировать элементы HTML-форм. Такая модульность позволяет использовать функцию снова и снова, если возникает необходимость повторно задействовать соответствующий элемент.

Создание динамической группы переключателей

Пусть нужно реализовать функциональность, которая с помощью следующего кода генерирует группу переключателей:

```
$option = array(“economy” => “Экономный”,
“standart” => “Стандартный”,
“express” => “Экспресс”);
```

```
$default = "economy";
$html = generate_radio_group("shipping", $options, $default);
```

Набор похожих функций значительно облегчает выполнение рутинных задач. Ниже приводится пример реализации функции generate_radio_group:

```
function_generate_radio_group ($name, $options, $default="") {
name = htmlentities($name);
foreach($options as $value => $label) {
$value = htmlentities($value);
$html .= "<INPUT TYPE=\"RADIO\" ";
if ($value == $default) {
  $html .= "CHECKED";
}
$html .= "NAME=\"$name\" VALUE=\"$value\">";
$html .= $label . "<br>";
}
return($html);
}
```

Порядок выполнения работы:

Упражнение 1. Установка значений по умолчанию.

Задание 1. Установка значений по умолчанию для текстового поля ввода.

```
<?
if (isset($_POST["quantity"])){
  $quantity=settype($_POST["quantity"],"integer"); }
else {
$quantity=1;}
$item_price=5.99;
printf("%d x item = $%.2f", $quantity, $quantity*$item_price);
?>
<FORM ACTION="buy.php" METHOD=POST>
Обновить количество:
<INPUT NAME="quantity" SIZE=2 VALUE="<?php echo $quantity;?>">
<INPUT TYPE=SUBMIT VALUE="Изменить количество">
</FORM>
```

Сценарий нужно сохранить в файле buy.php, тогда форма передаст данные в этот же сценарий. Если изменить количество и щелкнуть на кнопке подачи формы, сценарий рассчитает новую итоговую стоимость. После каждой перезагрузки страницы устанавливается новое значение по умолчанию.

Задание 2. Установка значений по умолчанию для группы переключателей.

```
<?php
if (!isset($shipping)) {
$shipping="economy";
}
echo "Вы получите $shipping вариант заказа";
?>

<FORM ACTION=shipping.php" METHOD=POST>
<INPUT TYPE="RADIO" NAME="shipping" VALUE="economy"
```

```
<?php if ($shipping==”economy”) echo “CHECKED”;?>>
Экономный
<INPUT TYPE=”RADIO” NAME=”shipping” VALUE=”standard”
<?php if ($shipping==”standard”) echo “CHECKED”;?>>
Стандартный
<INPUT TYPE=”RADIO” NAME=”shipping” VALUE=”express”
<?php if ($shipping==”express”) echo “CHECKED”;?>>
Быстрый
<INPUT TYPE=”SUBMIT” VALUE=”Изменить вариант заказа”>
</FORM>
```

Такой вариант выглядит не очень компактно даже для небольшой группы переключателей из трех элементов.Ниже показывается как динамически создать группу переключателей с помощью более изящного механизма.

Задание 3. Установка стандартного значения для раскрывающегося списка

```
<?php
if (!isset($shipping)) {
$shipping=”economy”;
}
echo “Вы получите $shipping вариант заказа”;
?>
<FORM ACTION=shipping.php” METHOD=POST>
<SELECT NAME=”shipping”>
<OPTION <?php if ($shipping==”economy”) echo “SELECTED”;?>
VALUE=”economy”>Экономный</OPTION>
<OPTION <?php if ($shipping==”standard”) echo “SELECTED”;?>
VALUE=”standard”>Стандартный</OPTION>
<OPTION <?php if ($shipping==”express”) echo “SELECTED”;?>
VALUE=”express”> Быстрый </OPTION>
<INPUT TYPE=”SUBMIT” VALUE=”Изменить вариант заказа”>
</FORM>
```

Как и для группы переключателей, с помощью функции динамической генерации можно создавать большие раскрывающиеся списки и задавать нужный стандартный вариант.

Упражнение 2. Создание элементов формы.

Задание 1. Реализация множественного выбора с помощью флажков

```
<?php
function generate_checkboxes($name, $options, $default=array()) {
  if (!is_array($default)) {
  $default=array();
  }
  foreach($options as $value => $label) {
  $html .= “<INPUT TYPE=CHECKBOX ”;
  if (in_array($value, $default)) {
  $html .=”CHECKED ”;
  }
  $html .= ”NAME=\”{$name}[]\” VALUE =\”$value\”>”;
```

```
   $html .= $label . "<br>";
   }
   return($html);
}
$option = array("movies" => "Ходить в кино",
                "music" => "Слушать музыку",
                "sport" => "Играть или смотреть спортивные
                  передачи",
                "travel" => "Путешествовать");
$html = generate_checkboxes("interests", $options, $interests);
?>
<H1>Пожалуйста, укажите область ваших интересов</H1>
<FORM ACTION="interests.php" METHOD=POST>
<?php print $html;?>
<INPUT TYPE=SUBMIT VALUE="Продолжить">
</FORM>
```

Задания для самостоятельной работы:

1. Создайте скрипт, генерирующий форму опроса на основе выше приве-денных примеров?

Правила отчетности студента:

Выполнить все лабораторные работы и показать преподавателю результаты работ полученные во время выполнения.

Контрольные вопросы:

1. Как создаются элементы формы?
2. Какие значения по умолчанию для текстового поля ввода задаются в атрибуте VALUE?

Список использованных источников:

1. Коггзолл, Джон. PHP 5. Полное руководство.: Пер. англ. – М: Издательский дом «Вильямс», 2006. – 750 с.: ил. – Парал. тит. англ.
2. Котеров Д. В. Самоучитель PHP 4. – СПб: БХВ-Петербург, 2001. – 576 с.: ил.
3. Ньюман, Крис. Освой самостоятельно PHP. 10 минут на урок. : Пер. англ. – М: Издательский дом «Вильямс», 2006. – 272 с. : ил. – Парал. тит. англ.

Список использованных источников:

1. Веллинг, Люк, Томсон, Лора., Разработка Web-приложений с помощью PHP и MySQL, 3-е издание: Пер. с англ.– М: Издательский дом «Вильямс», 2006. – 880 с.: ил. – Парал. тит. англ.
2. Бенкен Е.С. PHP, MySQL, XML: программирование для Интернета. – СПб: БХВ-Петербург, 2007. – 336 с.: ил. + CD-ROM
3. Коггзолл, Джон. PHP 5. Полное руководство.: Пер. англ. – М: Издательский дом «Вильямс», 2006. – 750 с.: ил. – Парал. тит. англ.
4. Котеров Д. В. Самоучитель PHP 4. – СПб: БХВ-Петербург, 2001. – 576 с.: ил.
5. Ньюман, Крис. Освой самостоятельно PHP. 10 минут на урок. : Пер. англ. – М: Издательский дом «Вильямс», 2006. – 272 с. : ил. – Парал. тит. англ.
6. Харрис Э. PHP/MySQL для начинающих./Пер. англ. – М: Кудиц-Образ, 2005. – 384 с.

Содержание

1. Общие указания к лабораторным работам ……...…... 3
2. Правила техники безопасности при выполнении лабораторных работ 4
3. Лабораторная работа № 1. Введение в программирование на PHP …...…. 5
4. Лабораторная работа № 2. Основы синтаксиса PHP ……...….…..... 10
5. Лабораторная работа № 3. Операторы PHP ……..…... 14
6. Лабораторная работа № 4. Управляющие операторы PHP 19
7. Лабораторная работа № 5. Работа со строками в PHP …...…….……... 25
8. Лабораторная работа № 6. Работа с формами в PHP ……...…….…….. 29
9. Лабораторная работа № 7. Обработка форм с помощью PHP ……..… 42
10. Лабораторная работа № 8. Динамическая генерация HTML-кода для форм …… 47
11. Список использованных источников ……..…… 52