Лабораторная работа № 27-28
Тема «Обработка запросов с помощью PHP»

Цель: рассмотреть основные понятия клиент-серверных технологий, охарактеризовать методы POST и GET, изучить механизм получения данных из HTML-форм и их обработки с помощью PHP. Проверить правильность выполнения программы создания формы для регистрации пользователей на сайте и отправке «универсального письма» всем зарегистрировавшимся. Разработать тестовую программу с отправкой результатов тестирования на сервер и их обработке с помощью PHP.

Время выполнения: 4 часа

1 Протокол http и способы передачи данных на сервер

Internet построен по многоуровневому принципу, от физического уровня, связанного с физическими аспектами передачи двоичной информации и до прикладного уровня, обеспечивающего интерфейс между пользователем и сетью.

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) 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).

Стоить отметить, что документации по протоколу HTTP огромное множество, при желании всегда можно более детально ознакомиться с ним самостоятельно.

2 Использование HTML-форм для передачи данных на сервер

Как передавать данные серверу? Для этого в языке HTML есть специальная конструкция – формы. Формы предназначены для того чтобы получать от пользователя информацию. Например, вам нужно знать логин и пароль пользователя для того, чтобы определить, на какие страницы сайта его можно допускать. Или вам необходимы личные данные пользователя, чтобы была возможность с ним связаться. Формы как раз и применяются для ввода такой информации. В них можно вводить текст или выбирать подходящие варианты из списка. Данные, записанные в форму, отправляются для обработки специальной программе (например, скрипту на PHP) на сервере. В зависимости от введенных пользователем данных эта программа может формировать различные web-страницы, отправлять запросы к базе данных, запускать различные приложения и так далее.

Разберемся с синтаксисом HTML-форм. Возможно, многие с ним знакомы, но все же повторим основные моменты, поскольку это важно.

Итак, для создания формы в языке HTML используется тег FORM. Внутри него находится одна или несколько команд INPUT. С помощью атрибутов action и method тега FORM задаются имя программы, которая будет обрабатывать данные формы, и метод запроса, соответственно. Команда INPUT определяет тип и различные характеристики запрашиваемой информации. Отправка данных формы происходит после нажатия кнопки input типа submit.

Создадим форму для регистрации участников спецкурсов по программированию

PHP-код программы и результаты его выполнения изображены на рисунке 31.

```
<h2> Форма для регистрации </h2>
<form action="31_1.php" method=POST><!--создаем форму-->
<!--данные формы будет обрабатывать файл31_1.phpпри
    отправке запроса будет использован метод POST-->
Имя <br><input type=text name="first_name"
     value="ВведитеВаше имя"><br>
Фамилия <br><input type=text name="last_name"><br>
E-mail <br><input type=text name="email"><br>
<p>
Выберите спецкурс, который вы хотели посещать: <br>
<input type=radio name="kurs" value="PHP">PHP<br>
<input type=radio name="kurs" value="Web-дизайн: HTML">Web-дизайн: HTML<br>
<input type=radio name="kurs" value="Web-дизайн: Flash">Web-дизайн: Flash<br>
<p> Что вы хотите, чтобы мы знали о вас? <br>
<textarea name="comment" cols=32 rows=5></textarea>
<p><input name="confirm" type=checkbox
checked>Подтвердить получение <br>
<input type=submit value="Отправить">
<input type=reset value="Отменитьь">
</form>
```

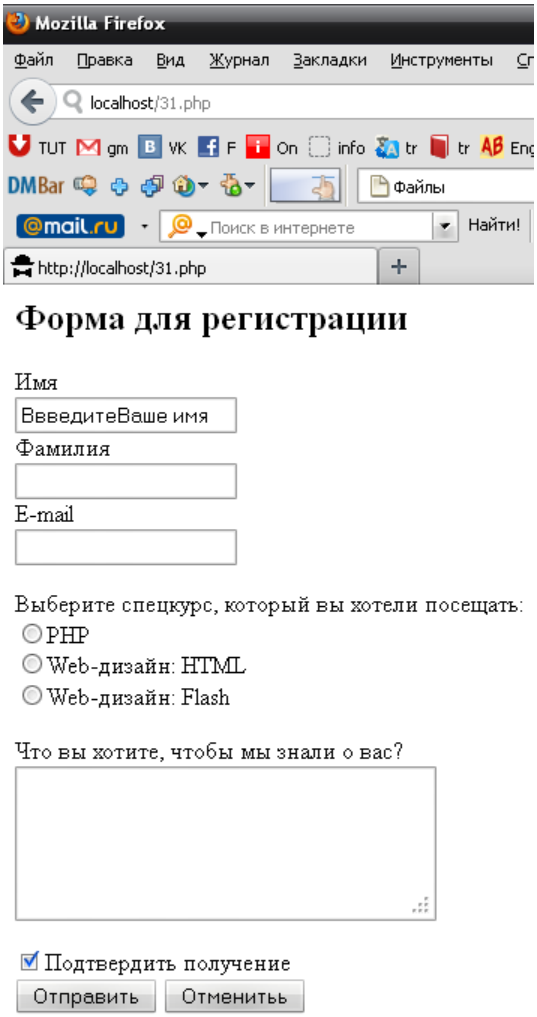


Рисунок 31

3 Метод 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?namel=valuel&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 namel, 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.

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.

4 Метод POST

Содержимое формы кодируется точно так же, как для метода GET (см. выше), 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.

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.

5 Обработка запросов с помощью PHP

До сих пор мы упоминали только, что запросы клиента обрабатываются на сервере с помощью специальной программы. На самом деле эту программу мы можем написать сами, в том числе и на языке PHP, и она будет делать с полученными данными все, что мы: захотим. Для того чтобы написать эту программу, необходимо познакомиться с некоторыми правилами и инструментами, предлагаемыми для этих целей PHP.

Внутри PHP-скрипта имеется 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 (напомним, что данные передаются в виде пар «имя переменной, символ «=», значение переменной»). 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, то есть его значения можно получить в любом месте программы, используя в качестве ключа имя соответствующей переменной (элемента формы).

Пример 31_1. Допустим создана форма для регистрации участников спецкурсов по программированию, как в предыдущем примере. Тогда в файле 31_1.php, обрабатывающем форму, можно записать следующее (рисунок: 31_1_а)

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

Рисунок 31_1_а

Если в форму ввести имя «Вася», фамилию «Петров» и выбрать среди всех курсов курс PHP, на экране браузера получим сообщение, изображенное на рисунке 31_1_б.

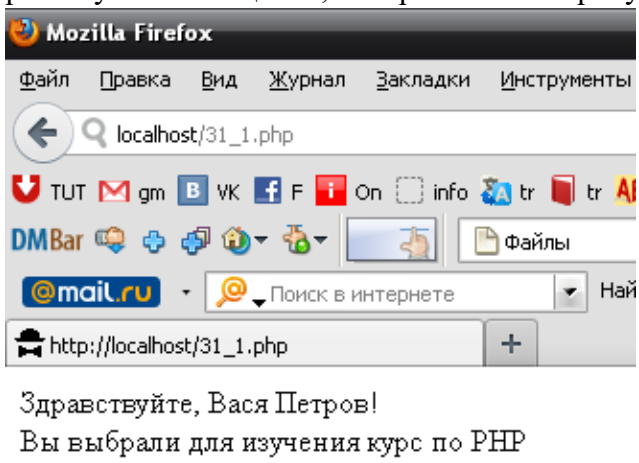


Рисунок 31_1_б

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, то есть допустимыми 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 другим и методами, обрабатывать не хотим. Это можно сделать следующим образом (рисунок 31_2_а).

```
1  <?php
2  $str="Здравствуйте,
3  ".$_POST["first_name"]."
4  ".$_POST["last_name"]."!<br>";
5  $str.="Вы выбрали для изучения курс по
6  ".$_POST["kurs"];
7  echo $str;
8  ?>
```

Рисунок 31_2_а

Тогда на экране браузера, если в форму ввести имя «Вася», фамилию «Петров» и выбрать среди всех курсов курс по Web-дизайн: HTML, на экране браузера получим сообщение, как в предыдущем примере (рисунок 31_2_б).

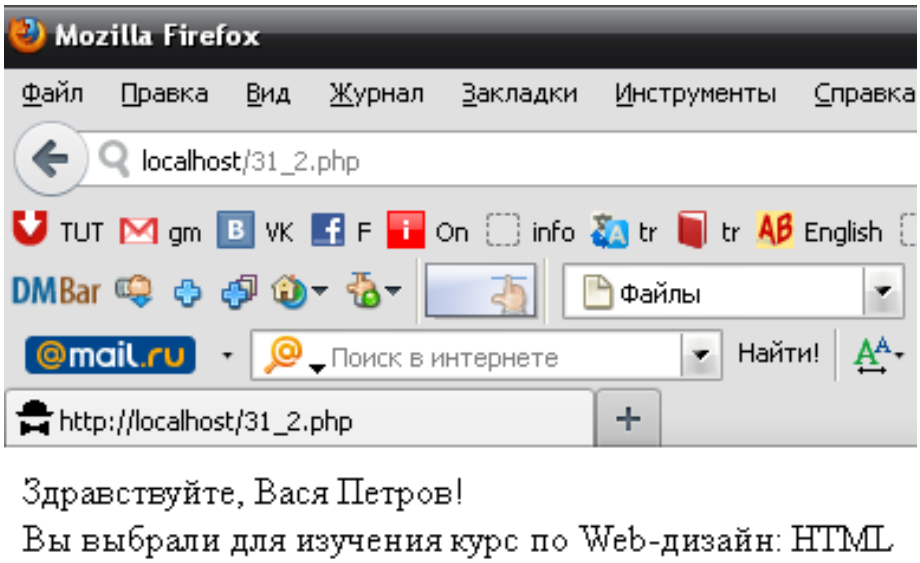


Рисунок 31_2_б

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.

PHP-код программы и результаты ее выполнения представлены на рисунке 31_3.

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

```

Рисунок 31_3.

6 Пример обработки запроса с помощью PHP

Необходимо написать форму для регистрации участников курсов по программированию и после регистрации отправить участнику сообщение. В примере не нужно отправляться что-либо по электронной почте, а просто необходимо сгенерировать сообщение и вывести его на экран браузера. Начальный вариант формы регистрации был приведен выше. Следует изменить его таким образом, чтобы каждый регистрирующийся мог выбрать сколько угодно курсов для посещения и не нужно подтверждать получение регистрационной формы.

HTML-код формы и ее вид представлены на рисунке 32.

```html
<h2> Форма для регистрации студентов</h2>
<form action="32_1.php" method=POST>
Имя <br><input type=text name="first_name"
     value="ВведитеВаше имя"><br>
Фамилия <br><input type=text name="last_name"><br>
E-mail <br><input type=text name="email"><br>
<p>
Выберите спецкурс, который вы хотели посещать: <br>
<input type=checkbox name="kurs[]" value="PHP">PHP<br>
<input type=checkbox name="kurs[]" value="Flash">Web-дизайн: Flash<br>
<input type=checkbox name="kurs[]" value="Oracle">Работа с Oracle SQL и PL/SQL<br>
<input type=checkbox name="kurs[]" value="HTML">Web-дизайн: HTML<br>
<p> Что вы хотите, чтобы мы знали о вас? <br>
<textarea name="comment" cols=32 rows=5></textarea><br>
<input type=submit value="Отправить">
<input type=reset value="Отменитьь">
</form>
```

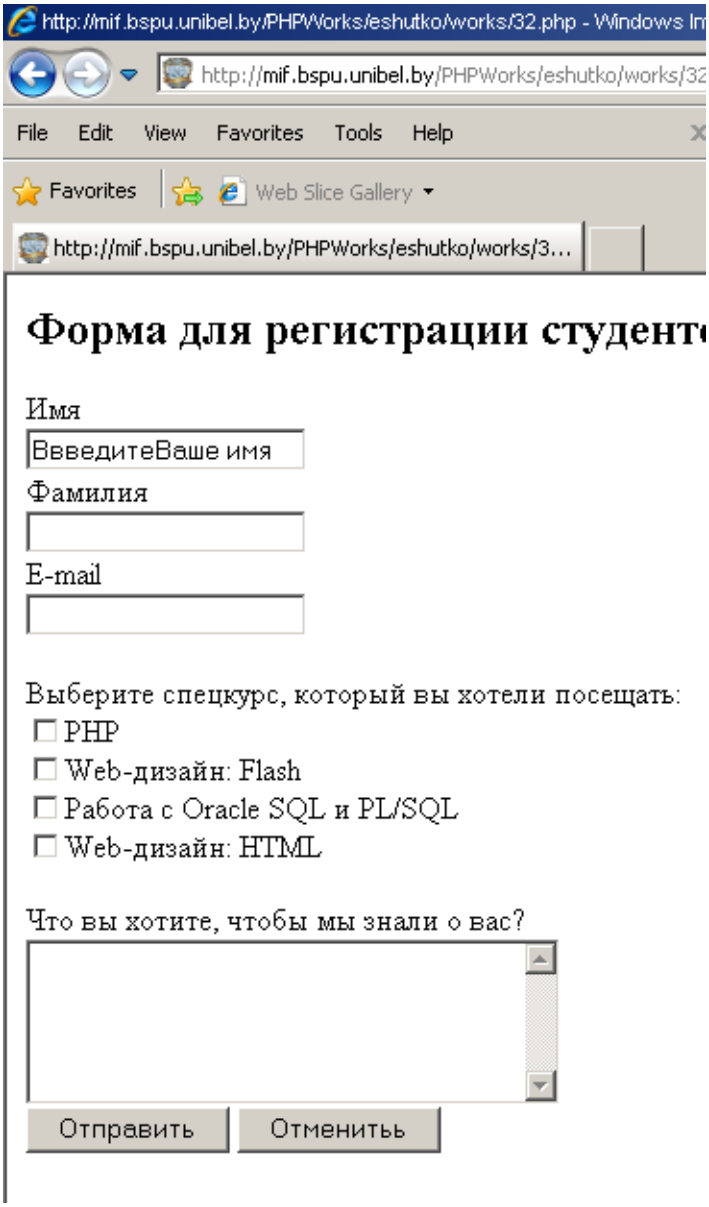


Рисунок 32

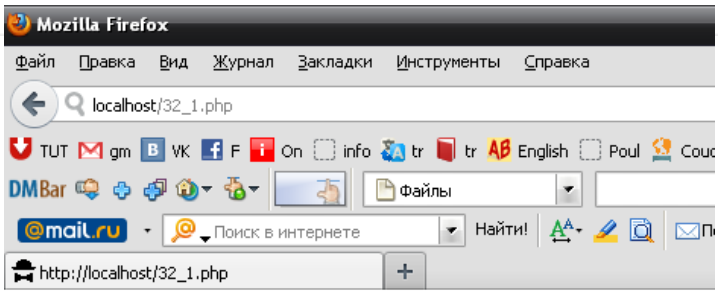
Здесь все достаточно просто и понятно. Единственное, что можно отметить, – это способ передачи значений элемента checkbox. Кода в имени элемента указывается kurs[], это значит, что первый отмеченный элемент checkbox будет записан в первый элемент массива kurs[], второй отмеченный checkbox – во второй элемент массива и так далее. Можно, конечно, просто дать разные имена элементам checkbox, но это усложнит обработку данных, если курсов будет много.

Скрипт, который все это будет обрабатывать, называется 32_1.php (форма ссылается именно на этот файл, что записано в ее атрибуте action). По умолчанию используется для передачи метод GET, но в программе указан POST. По полученным сведениям от зарегистрировавшегося студента, скрипт генерирует соответствующее сообщение. Если студент выбрал какие-либо спецкурсы, то ему выводится сообщение о времени их проведения и о лекторах, которые их читают. Если студент ничего не выбрал, то выводится сообщение о следующем собрании курсов.

PHP-код сценария и результат обработки формы представлены на рисунке 32_1

```php
<?php
//создадим массивы соответствий курс-время его
//проведения и курс-его лектор
$times=array (
       PHP=>"14.30",
       Flash=>"12.00",
       Oracle=>"15.00",
       HTML=>"14.00");
$lectors=array(
       PHP=>"Василий Васильевич",
       Flash=>"Иван Иванович",
       Oracle=>"Петр Петрович",
       HTML=>"Семен Семенович");
define ("SIGN","С уважением, администрация");
     //определим подпись письма как константу
define ("MEETING_TIME","18.00");
     //задаем время собрания студентов
$date="12 мая"; // задаем дату проведения лекций
     //начинаем составлять текст сообщения
$str="Здравствуйте, уважаемый " . $_POST["first_name"] . " " . $_POST["last_name"] .
"!<br>";
$str .="<br>Сообщаем Вам, что ";
$kurses=$_POST["kurs"]; // сохраняем в этой переменной
                                    // список выбранных курсов
$lect='';
if (!isset($kurses)) {// если не выбран ни один курс
    $event="следующее собрание ";
    $str .="$event состоится $date ". MEETING_TIME . "<br>";
} else {//если хотя бы один курс выбран
    $event="выбранные Вами лекции состоятся $date <ul>";
        // функция count вычисляет число элементов в массиве
    for ($i=0;$i<count($kurses);$i++){
                        //для каждого выбранного курса
         $k=$kurses[$i]; // запоминаем название курса
         $lect = $lect ."<li> лекция по $k  в $times[$k]";
                         //составляем сообщение
         $lect .="(Ваш лектор, $lectors[$k])";

    }
    $event = $event .$lect ."</ul>";
    $str .= "$event";
}
$str .="<br>". SIGN; //добавляем подпись
echo $str; // выводим сообщение на экран
?>
```



Здравствуйте, уважаемый Василий Васильев!

Сообщаем Вам, что выбранные Вами лекции состоятся 12 мая

- лекция по PHP в 14.30(Ваш лектор, Василий Васильевич)
- лекция по HTML в 14.00(Ваш лектор, Семен Семенович)

С уважением, администрация

Рисунок 32_1

Ниже представлен листинг программы.

```
<?php
//создали массивы соответствий курс-время его
//проведения и курс-его лектор
$times=array(
PHP=>"14.30",
CSS=>"12.00",
Oracle=>"15.00",
HTML=>"14.00");
$lectors=array(
PHP=>"Василий Васильевич",
Flash=>"Иван Иванович",
Oracle=>"Петр Петрович",
HTML=>"Семен Семенович");
define ("SIGN", "С уважением, администрация");
//определим подпись письма как константу
define("MEETING_TIME", "18.00");
//задаем время собрания студентов
$date="12 мая"; //задаем дату проведения лекций
//начинаем составлять текст сообщения
$str="Здравствуйте,уважаемый".$_POST["first_name"]. "
".$_POST["last_name"].
"!<br>";
$str.="<br>Сообщаем Вам, что";
$kurses=$_POST["kurs"];//сохраняем в этой переменной
//список выбранных курсов
$lect="";
if(!isset($kurses)){
$event="следующее собрание";
$str.="$event состоится $date". MEETING_TIME. "<br>";
}
else {
$event="выбранные Вами лекции состоятся $date <ul>";
for($i=0; $i<count($kurses); $i++){
$k=$kurses[$i];
$lect=$lect."<li> лекция по $k в $times[$k]";
$lect.="(Ваш лектор, $lectors[$k])";
}
$event=$event.$lect."</ul>";
$str.="$event";
}
$str.="<br>".SIGN;
echo $str;
?>
```

Итоги

Итак, рассмотрены ассоциативные массивы и основные методы, используемые для передачи данных на сервер, изучены средства, предлагаемые языком PHP для обработки клиентских запросов и испробованы на практике. В принципе этого уже достаточно, чтобы создавать клиент-серверные приложения.

ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

1 Изучить теоретический материал лабораторной работы.
2 Понять PHP-программы.
3 Отладить PHP-коды сценариев.
4 Выполнить индивидуальное задание.
5 Защитить лабораторную работу.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1 Объясните многоуровневый принцип построения Internet.
2 Поясните протокол передачи гипертекста HTTP.
3 Перечислите достоинства и недостатки метода GET при отправке данных с формы на сервер.
4 Перечислите достоинства и недостатки метода POST при отправке данных с формы на сервер.

5 Охарактеризуйте суперглобальный ассоциативный массив $_REQUEST.
6 Назовите основное различие в обработке запросов с помощью ассоциативных массивов $_GET и $_POST.
7 Объясните особенности обработки элемента checkbox с помощью PHP в примере обработки запроса из шестого раздела.

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

1 Разработать серверный сценарий обработки одностраничной и многостраничной тестовой формы.

ДОМАШНЯЯ РАБОТА

1. Изучить теоретический материал по учебному пособию [1, 388-396].
2. Отладить все рассмотренные примеры, подключить к сайту.
3. Письменно ответить на контрольные вопросы [1396].

ЛИТЕРАТУРА

1. Брылёва, А. А. Программные средства создания интернет-приложений : учеб. Пособие / А. А. Брылёва. – Минск : РИПО, 2022. – 483 с, : ил.

Лабораторная работа составлена
преподавателем спецдисциплин Еленой Иосифовной Шутько
март 2025 год