Тема: «HTTP-формы. Работа с формами методами GET и POST. Проверка передаваемых значений»

Цель: Научиться обрабатывать формы методами GET и POST

Теоретическая часть

Введение

PHP позволяет обрабатывать данные, которые пользователь ввел в поля формы. После активации кнопки *submit*данные отправляются на страницу – обработчик, указанную в поле *action* элемента <*form*>. На странице – обработчике располагается PHP скрипт, который выполняет определенные операции над полученными данными, например формирует и отсылает письмо по указанным пользователем реквизитам.

Передача данных обработчику

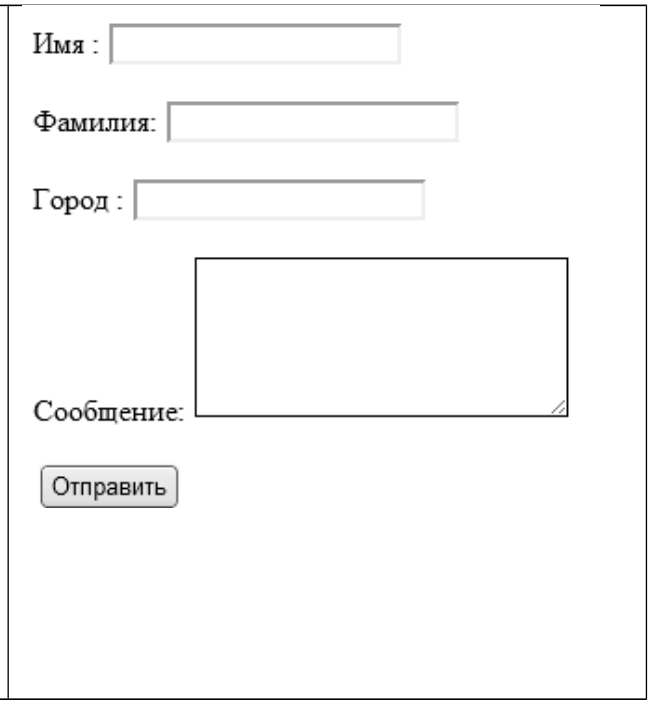
Данные из формы передаются на сервер как последовательность пар *имя/значение*. Это значит, что имя каждого элемента формы (появляющееся в атрибуте *NAME* тега) связывается со значением этого элемента (введеным или выбранным пользователем). Формат *имя/значение*, используемый для передачи, имеет вид *имя=значение*.

Все данные, передаваемые из формы в программу–обработчик располагаются в следующих суперглобальных массивах: *$_GET*, *$_POST*, или *$_REQUEST*.

$_GET[] — содержит все значения, передаваемые методом *GET*.
$_POST[] –содержит все значения, передаваемые методом *POST*.
$_REQUEST[] –содержит все значения, передаваемые методами *POST* и *GET*.

Пример работы формы:

```
<form action="process.php"
method="post">
Имя :
<input type="text"
name="FName"/>
Фамилия:
<input type="text"
name="LName"/>
Город :
<input type="text"
name="City"/>
Сообщение:
<textarea name="Message"
cols="30" rows="5">
</textarea>
<input type="submit"
name="submit"
value="Отправить"/>
</form>
```



После нажатия на кнопку *submit* данной формы все данные передаются обработчику *process.php*. Так как в этой форме используется метод *POST*, то все переменные будут расположенных внутри массива *$_POST*.

Теперь создадим обработчик:

```
process.php
<?php
echo "Имя: <font color="green"> " . $_POST["FName"] .
"</font><br/>";
echo "Фамилия: <strong> " . $_POST["LName"] . "</strong><br/>";
echo "Город: <em> " . $_POST["City"] . "</em><br/>";
echo "<br/>";
echo "Ваше сообщение : " . $_POST["Message"];
?>
```

В результате работы данный из формы передадутся на страницу *process.php.*

Массив $_Request

Использование суперглобального массива *$_Request* очень удобно, особенно когда не известно, каким методом были переданы данные.

Благодаря циклу *foreach* можно перебрать значения массива *$_Request*.

Пример:

```
<?php
foreach($_REQUEST as $key => $value)
{
echo $key;
echo ": ".$value;
echo "<br/>";
}
?>
```

В данном примере мы выводим на экран все значения массива *$_Request*. Сделано это может быть для проверки правильности ввода данных пользователем. То есть пользователь вводит данные в форму, нажимает отправить, но вместо обработки данных у него на экране высвечивается сообщение с введенными им данными и надписью подтвердить или отказаться. Данная идея применена на многих сайтах, да и вообще во многих программах.

Проверка корректности данных или допустимости вводимых данных

Данные, вводимые пользователем в форму и отправляемые в файл-обработчик, необходимо проверять на корректность.

Проверка на пустоту поля

Проверка того, что пользователь ввел данные, может осуществляться, к примеру, с помощью функции isset:

```
<?
$name = $HTTP_POST_VARS['name'];
if (!isset($name))
{ // если переменная $name не существует просим повторить ввод
имени
?>
<h1> Вы забыли ввести ваше имя </h1>
<!-- далее следует HTML-код формы, в которой вводится имя -->
<?  }
```

```
else  { … }
?>
```
Для этой же цели можно использовать функцию empty: <?
```
 $name = $HTTP_POST_VARS['name'];
  if (empty($name))
    {  // если поле пустое, снова просим ввести имя
?>
<h1> Вы забыли ввести ваше имя <h1>
<!-- далее следует HTML-код формы, в которой вводится имя -->
<?  }
   else  { … }
?>
```
На практике удобно сначала проверить, не пустой ли action формы, а потом уже проверять различные его составляющие: поле имя, e-mail и т. д. К примеру:
```
<?
$action = $HTTP_POST_VARS["action"];
if (!empty($action))
  {
   if (empty($name))
     {
   // код, для случая, когда не введено имя
     }
   if (!empty($email))
     {
   // код, для случая, когда не введен e-mail
     }
  // дальнейший код скрипта
  }
if (empty($action))
    {
?>
    <!-- здесь пишем HTML-код формы, в которой вводится информация -->
<? } ?>
```
Другие виды проверки:
Проверка данных электронной почты (длина сообщения, корректность адреса, фильтрация данных – удаление HTMLтегов и др.).
Для этих целей PHP имеет ряд функций:

- ***is_string()*** – позволяет проверить, является ли переменная строкой.
- ***is_int()*** – позволяет определить, является ли переменная целым числом.
- ***is_numeric()*** – позволяет определить, является ли переменная числом.
- ***is_numeric()*** – позволяет определить, является ли переменная числом с плавающей точкой.
- ***strlen(string)*** – позволяет определить длинну строки.
- ***strtolower()*** –преобразует все символы строки в нижний регистр.
- ***strtoupper(string)*** —преобразует все символы строки в верхний регистр.

Пример:

```
<?php
if ($_POST['submitB'] == "Submit")
{
```

```
$valid_form = true;

if ($_POST['name'] == "")
{
echo "Введите свое имя";
$valid_form = false;
}
else
{
$name = $_POST['name'];
}
if ($_POST['sname'] == "")
{
echo "Введите фамилию ";
$valid_form = false;
}
else
{
$sname = $_POST['sname'];
}
if ($_POST['pass'] == "")
{
echo "Введите пароль";
$valid_form = false;
}
elseif (strlen($_POST['pass']) < 6)
{
echo "Пароль должен содержать не менее 6 символов";
$valid_form = false;
}
else
{
$password = $_POST['pass'];
}
if($valid_form == true)
{
echo "Все поля формы заполненны корректно. Приветствуем вас
$name $sname <br>
Вы авторизовались под паролем $password<br><br>";
}
}
?>
```

Если обработчик обнаруживает ошибку, то просит пользователя исправить ее, выдавая соответствующее сообщение.

Обработка данных, введенных пользователем, позволяет избежать разнообразных ошибок, связанных с некорректными данными, введенными в поля форм.

Практическая часть

Задание 1

Создайте форму с элементами, показанными в таблице1.

Имя	
Фамилия	
Город	
Сообщение	
Отправить сообщение	Отправить

Таблица 1. Элементы формы.

Границы таблицы можно удалить.

Задание 2.

Создайте обработчик формы с выводом данных, как показано в таблице 2.

Имя	Александр
Фамилия	Константинов
Город	Волоколамск
Сообщение	PHP является молодым и динамично развивающимся языком программирования.

Таблица 2. Вывод данных из формы после обработки.

Задание 3.

Создайте проверку на заполнение данных для каждого элемента формы:
Имя, Фамилия, Город, Сообщение. Проверьте их действие.

Задание 4.

Спросите у пользователя логин и пароль (в браузере должен быть звездочками). Сравните их с логином **$login** и паролем **$pass**, хранящихся в файле. Если все верно — выведите *'Доступ разрешен!'*, в противном случае — *'Доступ запрещен!'*. Сделайте так, чтобы скрипт обрезал концевые пробелы в строках, которые ввел пользователь.