

Лекция 7

Передача данных. Методы GET
и POST в PHP. Session и Cookies

Метод GET в PHP

Веб-браузер связывается с сервером, как правило, с помощью одного из двух HTTP-методов (протокола передачи гипертекста) — GET и POST. Оба метода передают информацию по-разному и имеют разные преимущества и недостатки

В методе GET данные отправляются в виде параметров URL, которые обычно представляют собой строки пар имени и значения, разделенные амперсандами (&).

Метод GET в PHP

При использовании метода GET все переменные и их значения передаются прямо через адрес.

```
<html>
<head>
<title>Страница с примером
передачи переменных с
помощью Get</title>
</head>
<body>
<a
href=http://test/index1.php?
name=Alina&age=25>ссылка</a>
</body>
</html>
```

http://test/index1.php — адрес домена или, как его еще называют, хост.

index1.php — страница на php, которая будет обрабатывать запрос.

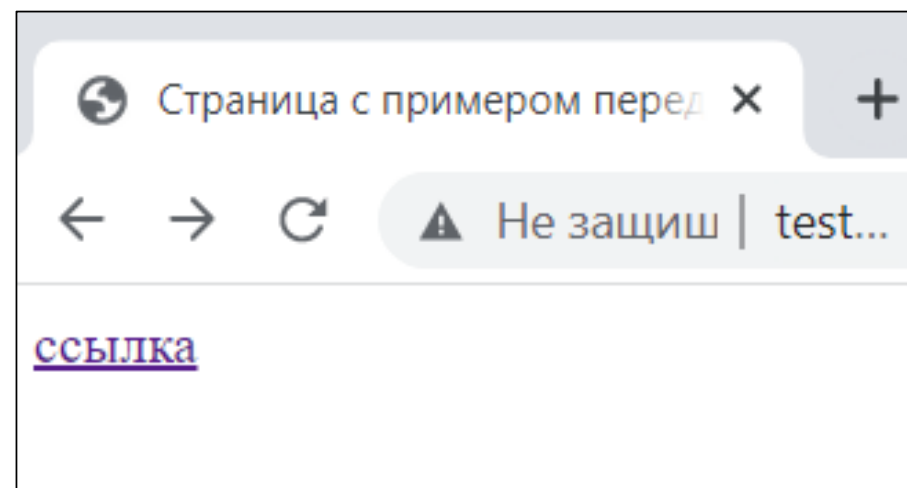
? — символ разделения между адресом и блоком с переменными.

Переменные и их значения, которые разделены символом *&* (*name = Alina и age = 25*).

Метод GET в PHP

При использовании метода GET все переменные и их значения передаются прямо через адрес.

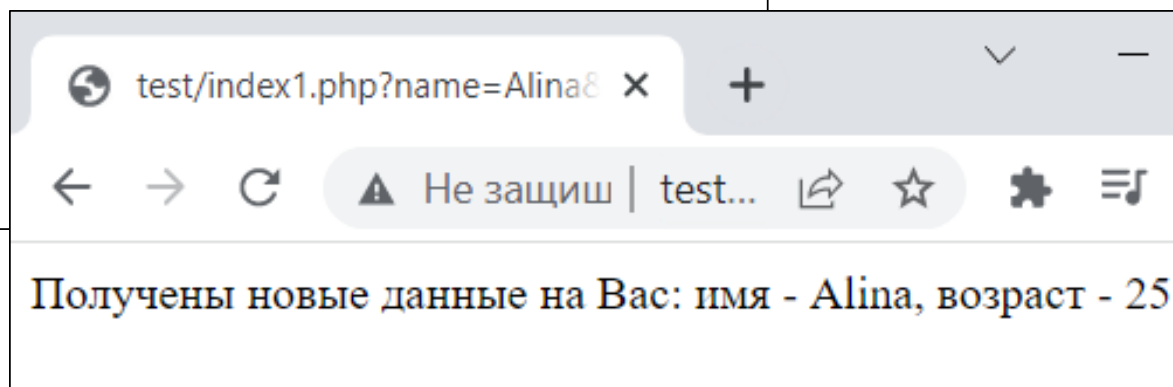
```
<html>
<head>
<title>Страница с примером
передачи переменных с
помощью Get</title>
</head>
<body>
<a
href=http://test/index1.php?
name=Alina&age=25>ссылка</a>
</body>
</html>
```



Метод GET в PHP

Скрипт index1.php, который будет выполнять обработку.

```
<?php
if ((!empty($_GET["name"])) && (!empty($_GET["age"])))
{
    echo "Получены новые данные на Вас: имя - " . $_GET["name"] . ",
возраст - " . $_GET["age"];
}
else
{
    echo "Данные на Вас не дошли";
}
?>
```



Метод GET в PHP

Преимущества и недостатки использования метода GET

- Поскольку данные, отправленные методом GET, отображаются в URL-адресе, можно добавить страницу в закладки с определенными значениями строки запроса.
- Метод GET не подходит для передачи конфиденциальной информации, такой как имя пользователя и пароль, поскольку они видны в строке запроса URL, а также потенциально хранятся в памяти браузера клиента в качестве посещенной страницы.
- Поскольку метод GET назначает данные переменной среды сервера, длина URL-адреса ограничена, т.е. существует ограничение на отправку общих данных.
- GET не может использоваться для отправки на сервер двоичных данных, таких как изображения или текстовые документы.

Метод POST в PHP

Метод POST передает информацию через HTTP-заголовки. Данные, отправленные с помощью метода POST, не будут отображаться в URL-адресе.

Для демонстрации работы этого метода нам понадобится немного больше, чем простая строка с адресом. Нужно будет создать html-страницу с формой для заполнения.

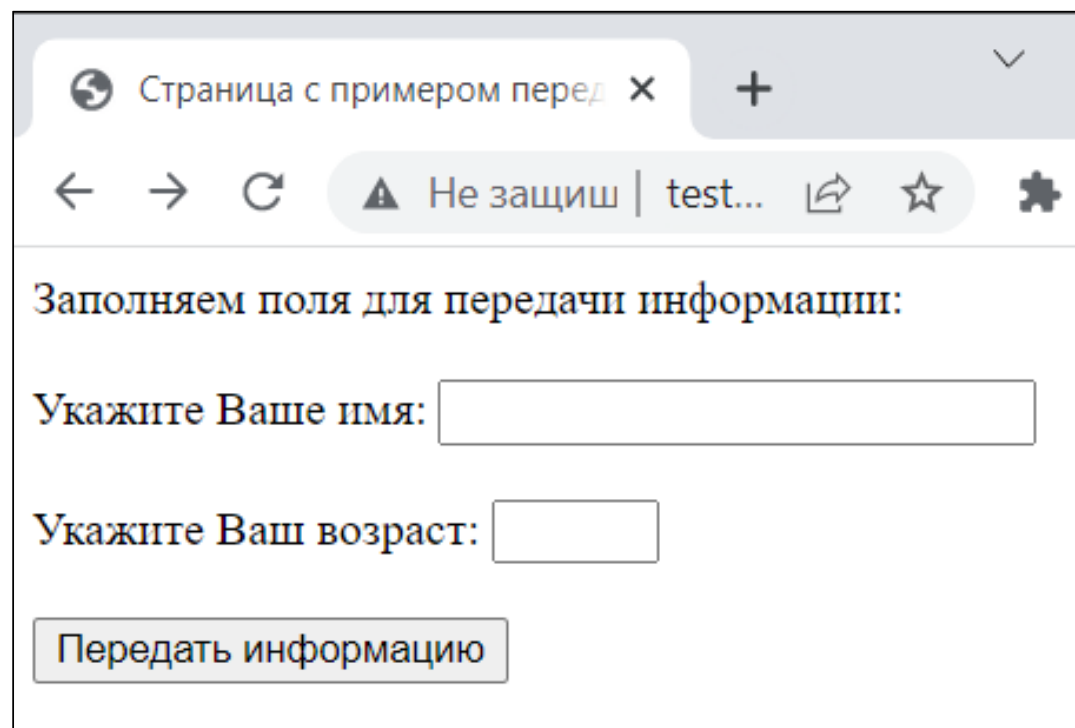
Метод POST в PHP

Страница primer.html

```
<html>
<head>
<title>Страница с примером передачи переменных с помощью Post</title>
</head>
<body>

<form method="post" action="index1.php">Заполняем поля для передачи информации:
<br><br>
Укажите Ваше имя:
<input name="user_name" type="text" maxlength="20" size="25" value="" />
<br><br>
Укажите Ваш возраст:
<input name="age" type="text" maxlength="2" size="3" value="" />
<br><br>
<input type="submit" value="Передать информацию"></form>
</body>
</html>
```


Метод POST в PHP



Страница с примером перед x +

← → ↻ ⚠ Не защищ | test... 🔗 ☆ ⚙

Заполняем поля для передачи информации:

Укажите Ваше имя:

Укажите Ваш возраст:

Метод POST в PHP

Первый параметр формы — «method», он определяет метод, который мы будем использовать для передачи. Это либо GET, либо POST. При этом, если установлен GET, то все имена полей (в виде названий переменных), а также их значения, передаются по ссылке, как в разделе про метод GET. Если же установлен POST, то все названия переменных и значения будут передаваться как запрос браузера к веб-серверу. То есть в адресной строке их видно не будет. Во многих случаях это очень полезно. Также POST безопаснее.

Второй параметр формы — «action». Это путь и имя файла скрипта, которому мы передаем данные. В нашем случае это index1.php. Этот путь можно передавать и полностью, то есть так: action=«http://test/index1.php». Если не указать значение параметра «action», то вся информация будет передаваться главному скрипту, то есть индексной странице index.php вашего сайта.

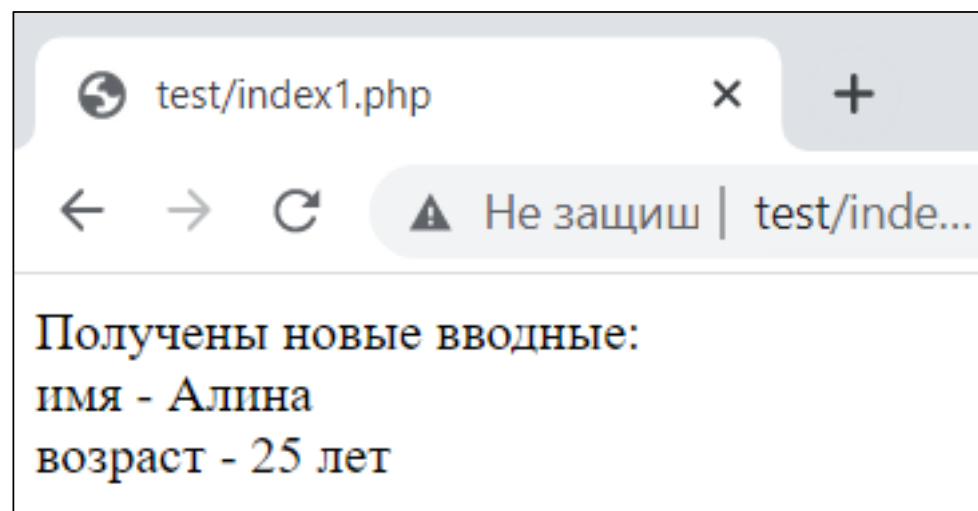
Метод POST в PHP

Скрипт index1.php, который будет выполнять обработку (empty — проверяет, пуста ли переменная).

```
<?php
if (!empty($_POST["user_name"]) && !empty($_POST["age"]))
{
    echo "Получены новые вводные:<br>";
    echo "имя - ";
    echo $_POST["user_name"];
    echo "<br>возраст - ";
    echo $_POST["age"];
    echo " лет";
}
else
{
    echo "Переменные не дошли. Проверьте все еще раз.";
}
?>
```

Метод POST в PHP

Результат



Метод POST в PHP

Страница primer.html

```
<html>
<head>
<title>Страница с примером передачи переменных с помощью Post</title>
</head>
<body>

<form method= "get" action="index1.php">Заполняем поля для передачи информации:
<br><br>
Укажите Ваше имя:
<input name="user_name" type="text" maxlength="20" size="25" value="" />
<br><br>
Укажите Ваш возраст:
<input name="age" type="text" maxlength="2" size="3" value="" />
<br><br>
<input type=submit value="Передать информацию"></form>
</body>
</html>
```

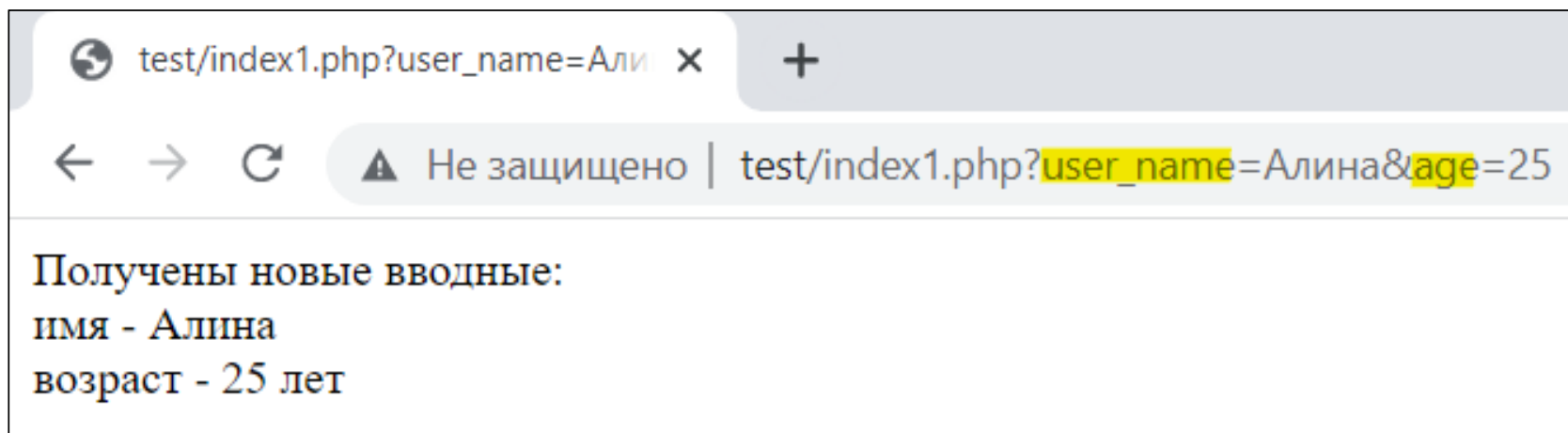
Метод POST в PHP

Скрипт index1.php, который будет выполнять обработку (empty — проверяет, пуста ли переменная).

```
<?php
if (!empty($_GET["user_name"]) && !empty($_GET["age"]))
{
    echo "Получены новые вводные:<br>";
    echo "имя - ";
    echo $_GET["user_name"];
    echo "<br>возраст - ";
    echo $_GET["age"];
    echo " лет";
}
else
{
    echo "Переменные не дошли. Проверьте все еще раз.";
}
?>
```

Метод POST в PHP

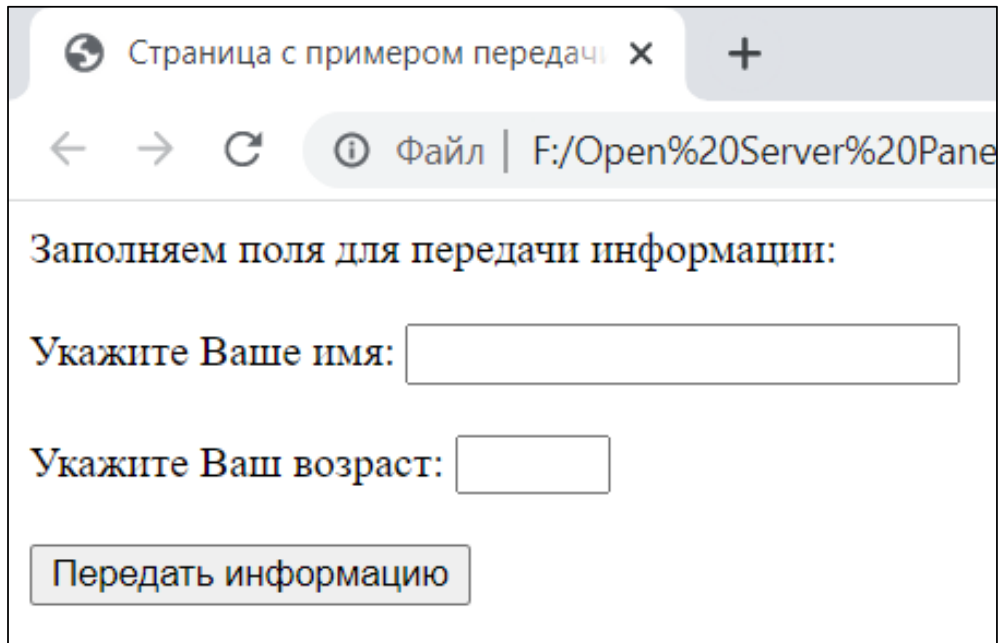
Результат



ВАЖНО: если в форме не указать `method`, по умолчанию метод передачи – GET.

Метод POST в PHP

ВАЖНО! Если вы откроете html-файл по локальной ссылке, а не через web-сервер, то скрипт выполняться не будет, а просто отобразится!!!

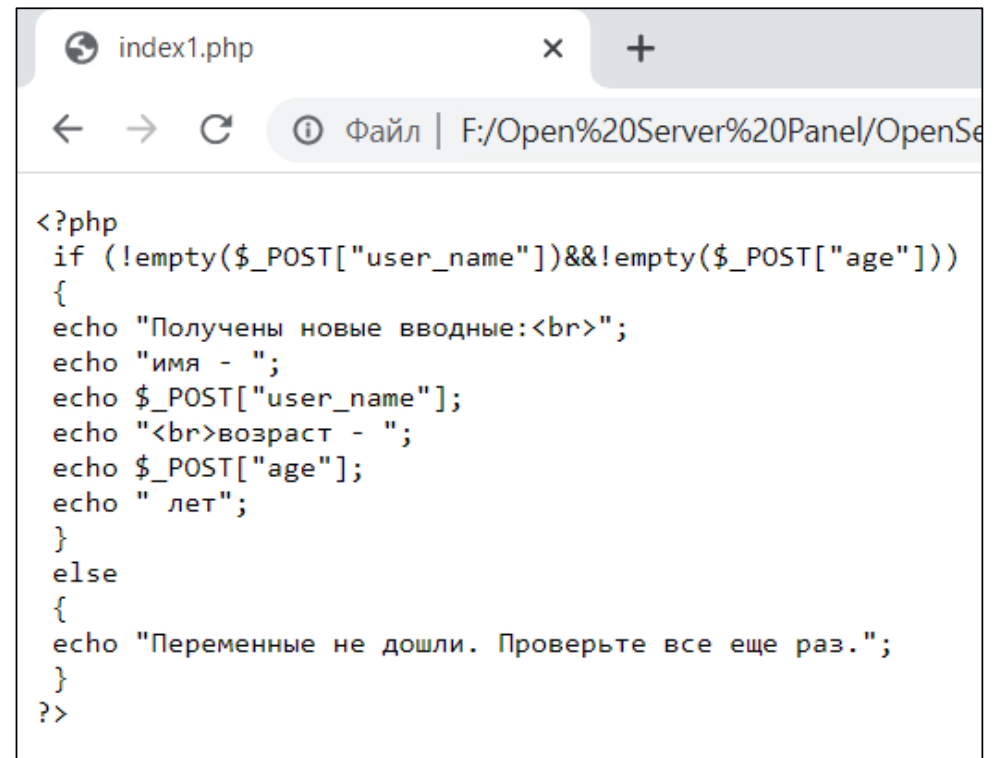


Страница с примером передачи информации

Заполняем поля для передачи информации:

Укажите Ваше имя:

Укажите Ваш возраст:



```
<?php
if (!empty($_POST["user_name"])&&!empty($_POST["age"]))
{
    echo "Получены новые вводные:<br>";
    echo "имя - ";
    echo $_POST["user_name"];
    echo "<br>возраст - ";
    echo $_POST["age"];
    echo " лет";
}
else
{
    echo "Переменные не дошли. Проверьте все еще раз.";
}
?>
```


Метод POST в PHP

Преимущества и недостатки использования метода POST

- Он более безопасен, чем GET, поскольку введенная пользователем информация никогда не отображается в строке запроса URL или в журналах сервера.
- Существует гораздо больший лимит на объем передаваемых данных; с помощью POST можно отправлять как текстовые, так и двоичные данные (загрузка файла).
- Поскольку данные, отправленные методом POST, не отображаются в URL-адресе, невозможно добавить страницу в закладки с помощью определенного запроса.

Метод POST в PHP

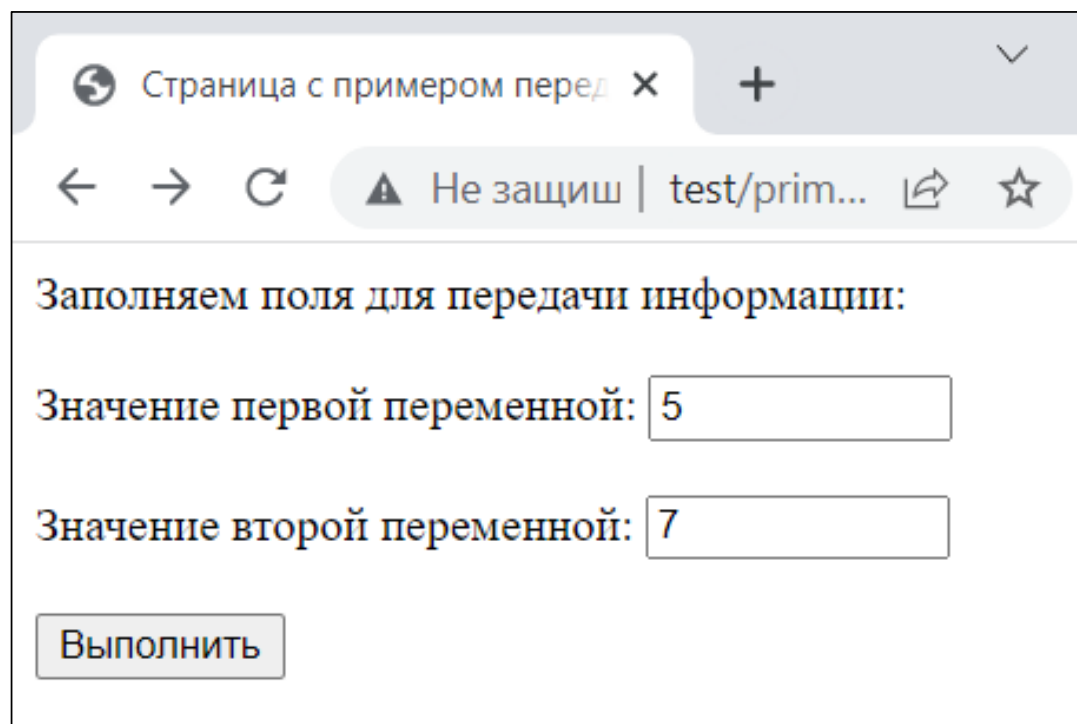
Если необходимо, чтобы результаты каких-либо операций выводились не просто на страницу, а в какое-либо поле, например, текстовое, то можно поступить следующим образом.

Страница с формой исходных данных и кнопкой вызова скрипта.

```
<html>
<head>
  <title>Страница с примером передачи переменных с помощью Post</title>
</head>
<body>

<form method="post" action="index1.php">Заполняем поля для передачи информации:
  <br><br>
  Значение первой переменной:
  <input name="A" type="text" maxlength="20" size="10" value="" />
  <br><br>
  Значение второй переменной:
  <input name="B" type="text" maxlength="20" size="10" value="" />
  <br><br>
  <input type="submit" value="Выполнить"></form>
</body>
</html>
```

Метод POST в PHP



Страница с примером перед

← → ↻ ⚠ Не защищ | test/prim... 🔗 ☆

Заполняем поля для передачи информации:

Значение первой переменной:

Значение второй переменной:

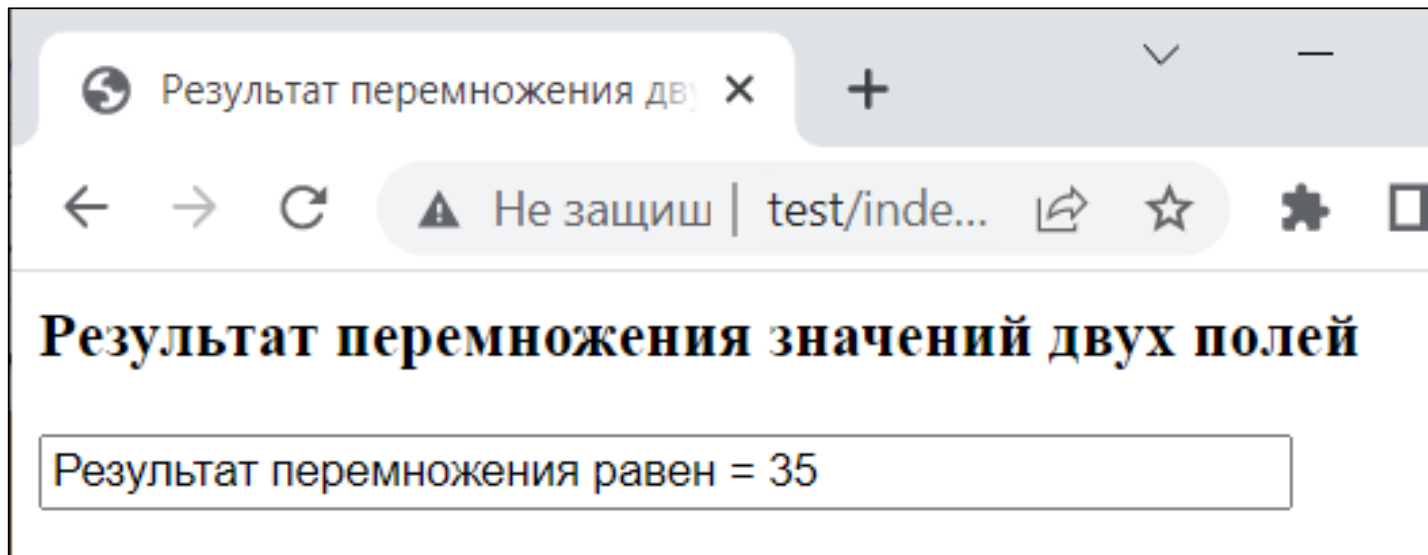
Метод POST в PHP

Непосредственно сам скрипт index1.php.

```
<html>
<head>
    <title>Результат перемножения двух чисел с выводом в текстовое окно</title>
</head>
<body>
<p><h3>Результат перемножения значений двух полей</h3></p>
<?php
if (!empty($_POST["A"]) && !empty($_POST["B"]))
{
    $C=$_POST["A"]*$_POST["B"];
    echo "<input type='text' name='company' size = '50' value = 'Результат
перемножения равен = {$C}' />";
}
else
{
    echo "<input type='text' name='company' size = '50' value = 'Переменные не
дошли. Проверьте все еще раз.' />";
}
?>
</body>
</html>
```

Метод POST в PHP

Результат



Элементы HTML-форм

1. Кнопки - задаются с помощью элементов BUTTON и INPUT. Различают:

- кнопки отправки - при нажатии на них, они осуществляют отправку формы серверу;
- кнопки сброса - при нажатии на них, управляющие элементы принимают первоначальные значения;
- прочие кнопки - кнопки, для которых не указано действие, выполняемое по умолчанию при нажатии на них.

2. Зависимые переключатели (переключатели с зависимой фиксацией) - задаются элементом INPUT и представляют собой переключатели "вкл/выкл". Если несколько зависимых переключателей имеют одинаковые имена, то они являются взаимоисключающими. Это значит, что если одна из них ставится в положение "вкл", то все остальные автоматически - в положение "выкл". Именно это и является преимуществом их использования.

Элементы HTML-форм

3. Независимые переключатели (переключатели с независимой фиксацией) - задаются элементом INPUT и представляют собой переключатели "вкл/выкл", но в отличие от зависимых, независимые переключатели могут принимать и изменять свое значение независимо от остальных переключателей. Даже если последние имеют такое же имя.
4. Меню - реализуется с помощью элементов SELECT, OPTGROUP и OPTION. Меню предоставляют пользователю список возможных вариантов выбора.
5. Ввод текста - реализуется элементами INPUT, если вводится одна строка, и элементами TEXTAREA - если несколько строк. В обоих случаях введенный текст становится текущим значением управляющего элемента.
6. Выбор файлов - позволяет вместе с формой отправлять выбранные файлы, реализуется HTML-элементом INPUT.
7. Скрытые управляющие элементы - создаются управляющим элементом INPUT.

Тег form

Тег FORM своими атрибутами указывает адрес сценария (скрипта), которому будет послана форма, способ пересылки и характеристику данных, содержащихся в форме. Начальный и конечный теги FORM задают границы формы, поэтому их указание является обязательным.

Атрибуты тега FORM:

- **action**- единственный обязательный атрибут. В качестве его значения указывается URL-адрес запрашиваемого скрипта, которая будет обрабатывать данные, содержащиеся в форме. Допустимо использовать запись `mailto:URL`, благодаря которой форма будет послана по электронной почте. Если атрибут ACTION все-таки не указан, то содержимое формы будет отправлено на URL-адрес, с которого загружалась данная веб-страница;
- **method** - определяет метод HTTP, используемый для пересылки данных формы от браузера к серверу. Атрибут METHOD может принимать два значения: GET и POST;
- **enctype** - необязательный атрибут. Указывает тип содержимого формы, используемый для определения формата кодирования при ее пересылке.

Тег input

Элемент INPUT является наиболее употребительным тегом HTML-форм. С помощью этого тега реализуются основные функции формы. Он позволяет создавать внутри формы поля ввода строки текста, имени файла, пароля и.т.д.

Особенность INPUT - нет конечного (завершающего) тега. Атрибуты и особенности использования INPUT зависят от способа его использования.

Однострочные поля ввода

```
<input type=text name=имя_параметра [value=значение]  
[size=размер_поля] [maxlen=длина_поля]>
```

Данный тег создает поле ввода с максимально допустимой длиной текста **maxlen** и размером в **size** знакомест. Если указан атрибут **value**, то в поле будет изначально отображаться значение данного атрибута. В квадратных скобках [] помечены необязательные атрибуты.

Тег input

Поле ввода пароля

```
<input type=password name=имя_параметра [value=значение]  
[size=размер] [maxlen=длина]>
```

Скрытое текстовое поле

```
<input type=hidden name=имя_параметра value=значение>
```

Независимые переключатели

```
<input type=checkbox name=имя_параметра value=значение [checked]>
```

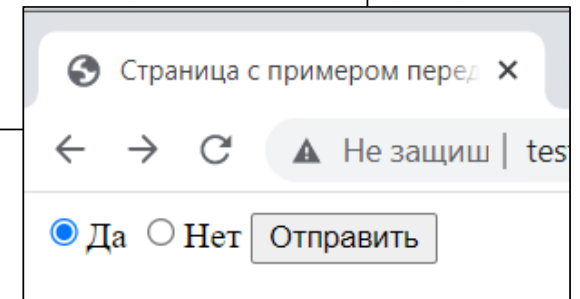
Если переключатель был включен на момент нажатия кнопки отправки данных, то скрипту будет передан параметр имя=значение. Если же флажок выключен, то сценарию вообще ничего не будет передано - как будто переключателя вообще нет.

Переключатель по умолчанию либо включен, либо выключен. Чтобы переключатель был по умолчанию включен, необходимо для него указать атрибут **checked**.

Тег input

Зависимые переключатели

```
<form action="http://localhost/script.php" method="GET">  
<input type=radio name=answer value=yes checked>Да  
<input type=radio name=answer value=no>Нет  
<input type=submit value=Отправить>  
</form>
```



Зависимый переключатель, также как и независимый переключатель, может быть либо включен, либо выключен. При этом переключатель **radio** является зависимым переключателем, поскольку на форме может быть только один включенный переключатель типа **radio**.

Тег input

Кнопка отправки формы

```
<input type=submit [name=go] value=Отправить>
```

Атрибут **value** определяет текст, который будет написан на кнопке отправки. Атрибут **name** определяет имя кнопки и является необязательным. Если значение этого атрибута не указывать, то скрипту будут переданы введенные в форму значения и все. Если атрибут **name** для кнопки будет указан, то дополнительно к основным данным формы будет отправлена пара **имя=значение** от самой кнопки.

Тег input

Кнопка сброса параметров

```
<input type=reset value=Сброс>
```

Кроме кнопки submit есть еще кнопка **reset**, которая сбрасывает параметры формы, а точнее, устанавливает для всех элементов формы значения по умолчанию.

Кнопка отправки с рисунком

```
<input type=image name=имя src=рисунок>
```

Вместо кнопки submit можно использовать рисунок для отправки данных. Клик на этом рисунке дает то же самое, что и нажатие на кнопку submit. Однако, кроме этого, сценарию будут переданы координаты места клика на рисунке. Координаты будут переданы в формате имя. x=коор_X, y=коор_Y.

Тег textarea

Многострочные текстовые поля. Тег TEXTAREA

Поле, создаваемое этим тегом, позволяет вводить и отправлять не одну строку, а сразу несколько строк. Синтаксис тега TEXTAREA:

```
<textarea name=имя [cols=ширина_в_символах]  
[rows=высота_в_символах] wrap=тип_переноса>  
текст по умолчанию  
</textarea>
```

Несколько значений относительно использования атрибутов: необязательные параметры **cols** и **rows** желательно все-таки указывать. Первый из них задает количество символов в строке, а второй - количество строк в области. Атрибут **wrap** определяет тип переноса текста, как будет выглядеть текст в поле ввода:

Virtual - справа от текстового поля выводится полоса прокрутки. Вводимый текст выглядит разбитым на строки, а символ новой строки вставляется при нажатии клавиши ENTER;

Physical - этот тип зависит от типа браузера и выглядит по-разному;

None - текст выглядит в поле в том виде, в котором пользователь его вводит. Если текст не умещается в одну строку, появляется горизонтальная полоса прокрутки.

Списки выбора. Тег select

Списки с единственным выбором

В HTML списки реализуются с помощью тега SELECT. Список выбора позволяет выбрать один вариант из множества.

```
<select name=day size=1>
<option value=1>Понедельник</option>
<option value=2>Вторник</option>
<option value=3 selected>Среда</option>
<option value=4>Четверг</option>
<option value=5>Пятница</option>
<option value=6>Суббота</option>
<option value=7>Воскресенье</option>
</select>
```

Атрибут **name** определяет имя параметра, который будут передан скрипту. Если атрибут **size** равен 1, то список будет "оснащен" полосой прокрутки. Значение, выбранное в списке по умолчанию, можно указать с помощью атрибута **selected** для соответствующего тега **option**. В приведенном примере день недели по умолчанию - Среда. Атрибут **value** является необязательным. Если его не указать, то будет передана строка, заключенная в тег **option**.

Списки выбора. Тег select

Списки множественного выбора

Для того, чтобы создать список с множественным выбором, необходимо для тега SELECT указать атрибут **multiple**.

```
<select name=day size=7 multiple>
<option value=1>Понедельник</option>
<option value=1>Вторник</option>
<option value=1>Среда</option>
<option value=1>Четверг</option>
<option value=1>Пятница</option>
<option value=1>Суббота</option>
<option value=1>Воскресенье</option>
</select>
```


Тег input

Загрузка файлов на сервер

Тег INPUT позволяет реализовать еще одну возможность форм, а именно создавать поле выбора файла для его отправки на сервер.

```
<input type=file name=имя [value=имя_файла]>
```

Функция isset()

```
<?php
$var = '';
// Проверка вернет TRUE, поэтому текст будет напечатан.
if (isset($var)) {
    echo "Эта переменная определена, поэтому меня и напечатали.";
}
// В следующем примере мы используем var_dump для вывода
// значения, возвращаемого isset().
$a = "test";
$b = "anothertest";
var_dump(isset($a));           // TRUE
var_dump(isset($a, $b));       // TRUE
unset ($a);
var_dump(isset($a));           // FALSE
var_dump(isset($a, $b));       // FALSE
$foo = NULL;
var_dump(isset($foo));         // FALSE
?>
```

Функция isset() проверяет, установлена ли переменная, что означает, что она должна быть объявлена и не равна NULL. Эта функция возвращает значение true, если переменная существует и не равна NULL, в противном случае она возвращает значение false.

Функция isset()

Также можно проверить, загружен ли сценарий методом POST, то есть была ли отправлена форма.

```
if (isset($_POST['checkbox1'])) echo $_POST['checkbox1'];  
if (isset($_POST['radiobutton'])) echo $_POST['radiobutton'];
```

Обработка параметров списка с множественным выбором

```
day_m=01&day_m=03&day_m=07...
```

Один параметр имеет несколько значений. Это напоминает массив данных. Действительно, множественный список можно представить в виде массива, а обработать его элементы с помощью цикла `foreach`. При этом даже не обязательно знать количество элементов множественного списка. Нам нужно лишь предварительно дать понять PHP, что мы будем передавать массив:

```
<select name="day_m[]" size=7 multiple>
```

Квадратные скобки `[]` - это признак массива. Циклическая обработка массива осуществляется так:

```
foreach ($_POST['day_m'] as $key=>$value) echo "$key = $value <br>";
```

Сессии в PHP

Сессия - это набор параметров и значений, однозначно определяющих пользователя. Каждая сессия обладает собственным уникальным идентификатором, поэтому пользователю достаточно сообщить серверу идентификатор своей сессии и сервер "вспомнит" все подробности и историю запросов пользователя.

Идентификатор сессии автоматически сохраняется в браузере пользователя в виде cookie, а если браузер cookie не поддерживает - идентификатор добавляется автоматически к адресу страницы и всем ссылкам на ней. Это значит, что при обновлении страницы браузер сам отправит на сервер идентификатор сессии, независимо от действий пользователя.

Сессии в PHP

При создании **PHP-сессии** выполняются следующие три действия:

Когда создается сессия, PHP генерирует уникальный идентификатор, который представляет собой случайную строку из 32 шестнадцатеричных чисел. Идентификатор времени жизни сессии PHP выглядит примерно так: **9c8foj87c3jj973actop1re472e8774**;

Сервер отправляет на компьютер пользователя cookie, называемые **PHPSESSID**, для хранения строки уникального идентификатора сессии;

Сервер генерирует в указанном временном каталоге файл, который содержит имя уникального идентификатора сессии с префиксом **sess_g. sess_9c8foj87c3jj973actop1re472e8774**.

Эти установки помогают скрипту PHP извлекать из файла значения переменных сессии. На стороне клиента **PHPSESSID** содержит идентификатор сессии. Он подтверждает имя файла, который нужно искать в определенном каталоге на стороне сервера, из него переменные сессии могут быть извлечены и использованы для проверки.

Сессии в PHP

Для создания новой сессии необходимо вызвать функцию `session_start()` без параметров, после этого станет доступен глобальный массив `$_SESSION` и в cookies браузера пользователя появится элемент `PHPSESSID`.

```
<?php
    session_start();
    $_SESSION['sampleName'] = 'Иван Иванович';
    echo $_SESSION['sampleName'];
?>
```

Сессии в PHP

Сессия иницируется один раз, а все повторные вызовы `session_start()` приводят только к появлению предупреждения "Сессия уже открыта, повторное открытие не допускается.". Для проверки, была ли уже вызвана функция `session_start()` существует специальная функция `session_id()`.

```
<?php
// открываем сессию, только если она ещё не была открыта ранее
    if(session_id() == '')
        session_start();
?>
```


Сессии в PHP

Для того, чтобы ваши данные сохранялись на сервере при переходах между страницами, достаточно поместить их в глобальный массив `$_SESSION`.

```
<?php
// открываем сессию
session_start();
// проверяем на наличие имени в URL-запросе
if(isset($_GET['name']))
{
    // получаем имя
    $name = $_GET['name'];

    // и сохраняем его в сессии
    $_SESSION['name'] = $name;
}
?>
```

Сессии в PHP

Иногда возникают ситуации, когда надо заново сгенерировать идентификатор сессии. Для этого служит функция `session_regenerate_id()`. Вызывать её нужно после открытия сессии:

```
<?php

if(session_id() == '')
    session_start();
else session_regenerate_id();

?>
```

Сессии в PHP

Элементы сессии могут быть не только простых типов (как было показано ранее), но и массивами:

```
<?php
    ...
    $_SESSION['myWorkWeek'] = array();
    ...
    $_SESSION['myWorkWeek']['monday'] = 'Понедельник';
    $_SESSION['myWorkWeek']['tuesday'] = 'Вторник'; ...
    $_SESSION['myWorkWeek']['friday'] = 'Пятница';
    echo 'Мои рабочие дни: ' .
    implode(', ', $_SESSION['myWorkWeek']);
    // выведет:
    // Мои рабочие дни: Понедельник, Вторник, ...
?>
```

Сессии в PHP

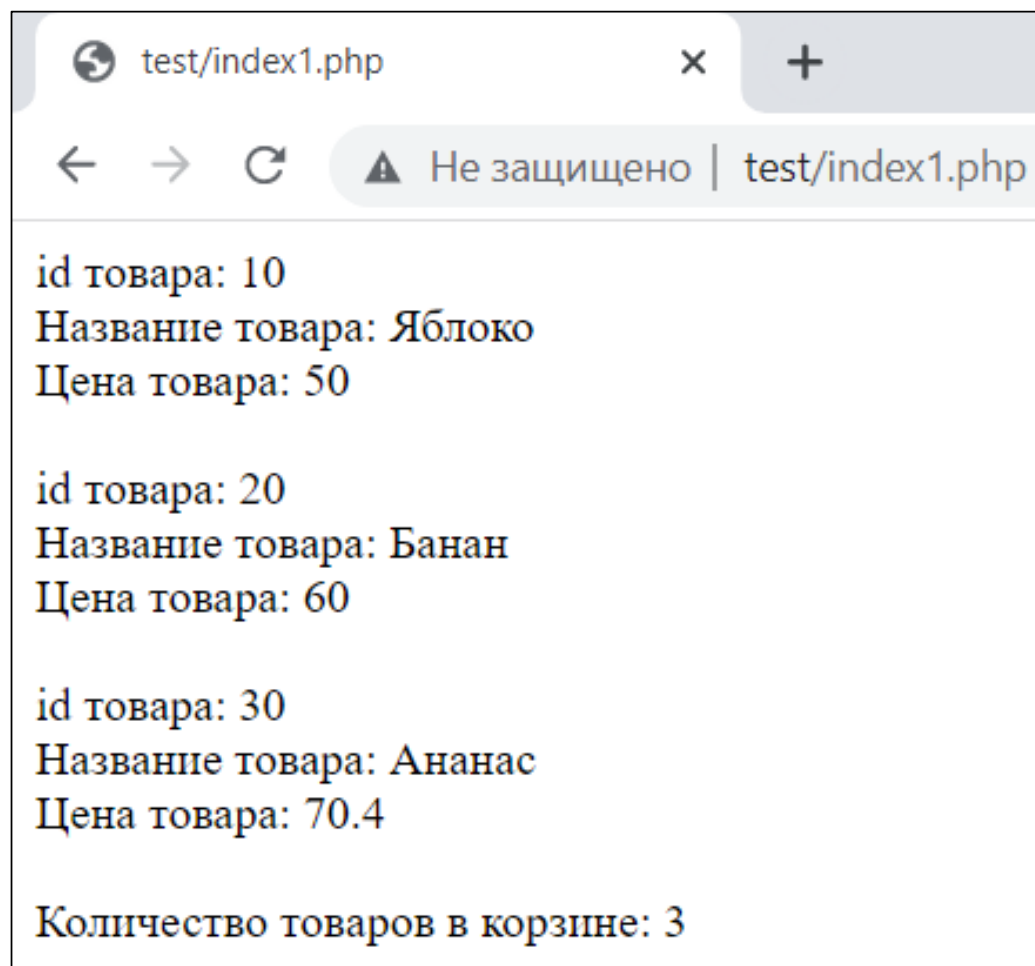
Рассмотрим пример:

у вас есть корзина, в ней хранятся продукты, необходимо это записать в сессию

```
<?php
$items = Array(
    0 => Array
        (
            "ID" => 10, "NAME" => "Яблоко", "PRICE" => 50),
    1 => Array
        (
            "ID" => 20, "NAME" => "Банан", "PRICE" => 60),
    2 => Array
        (
            "ID" => 30, "NAME" => "Ананас", "PRICE" => 70.4)
);
$_SESSION['items'] = $items;
foreach ($_SESSION['items'] as $item) {
    echo "id товара: ".$item['ID']."<br>";
    echo "Название товара: ".$item['NAME']."<br>";
    echo "Цена товара: ".$item['PRICE']."<br><br>";
}
echo "Количество товаров в корзине: ".count($_SESSION['items']);
?>
```

Сессии в PHP

Результат:



Сессии в PHP

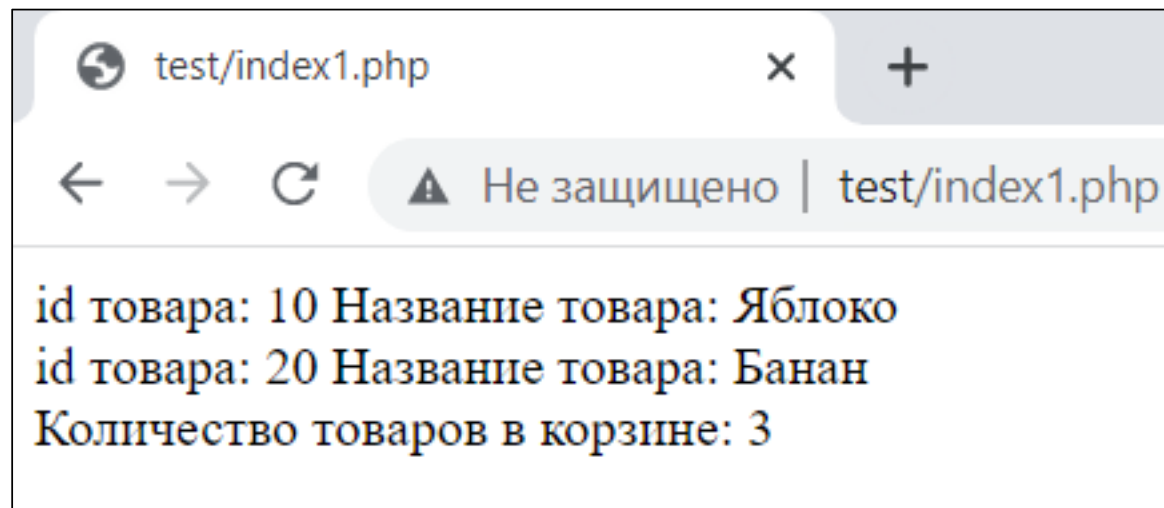
Если нужно обратиться к элементу массива напрямую, то это можно будет сделать так:

```
<?php
$items = Array(
    0 => Array
        ("ID" => 10, "NAME" => "Яблоко", "PRICE" => 50),
    1 => Array
        ("ID" => 20, "NAME" => "Банан", "PRICE" => 60),
    2 => Array
        ("ID" => 30, "NAME" => "Ананас", "PRICE" => 70.4)
);
$_SESSION['items'] = $items;
    echo "id товара: {$_SESSION['items'][0]['ID']} ";
    echo "Название товара: {$_SESSION['items'][0]['NAME']} <br>";
    echo "id товара: {$_SESSION['items'][1]['ID']}";
    echo "Название товара: {$_SESSION['items'][1]['NAME']}";

echo "Количество товаров в корзине: ".count($_SESSION['items']);
?>
```

Сессии в PHP

Результат:



Сессии в PHP

Сессию можно закрыть или уничтожить. Закрытие сессии происходит при вызове функции `session_write_close()`. При этом все изменения, сделанные в сессии, сохраняются в дисковом кеше. Если же необходимо закрыть сессию и удалить все её данные - надо вызвать `session_destroy()` или `session_unset()`. Отличие между ними в том, что `session_unset()` может использоваться в старом коде, где недоступен глобальный массив `$_SESSION`.

```
<?php
...
// освобождаем все переменные сессии
$_SESSION = array();
// стираем cookie идентификатора сессии
if (isset($_COOKIE[session_name()]))
{
    setcookie(session_name(), '', time()-42000, '/');
}
// уничтожаем сессию
session_destroy();
?>
```


Сессии в PHP

Рассмотрим пример: создадим 3 страницы page1.php, page2.php, page3.php

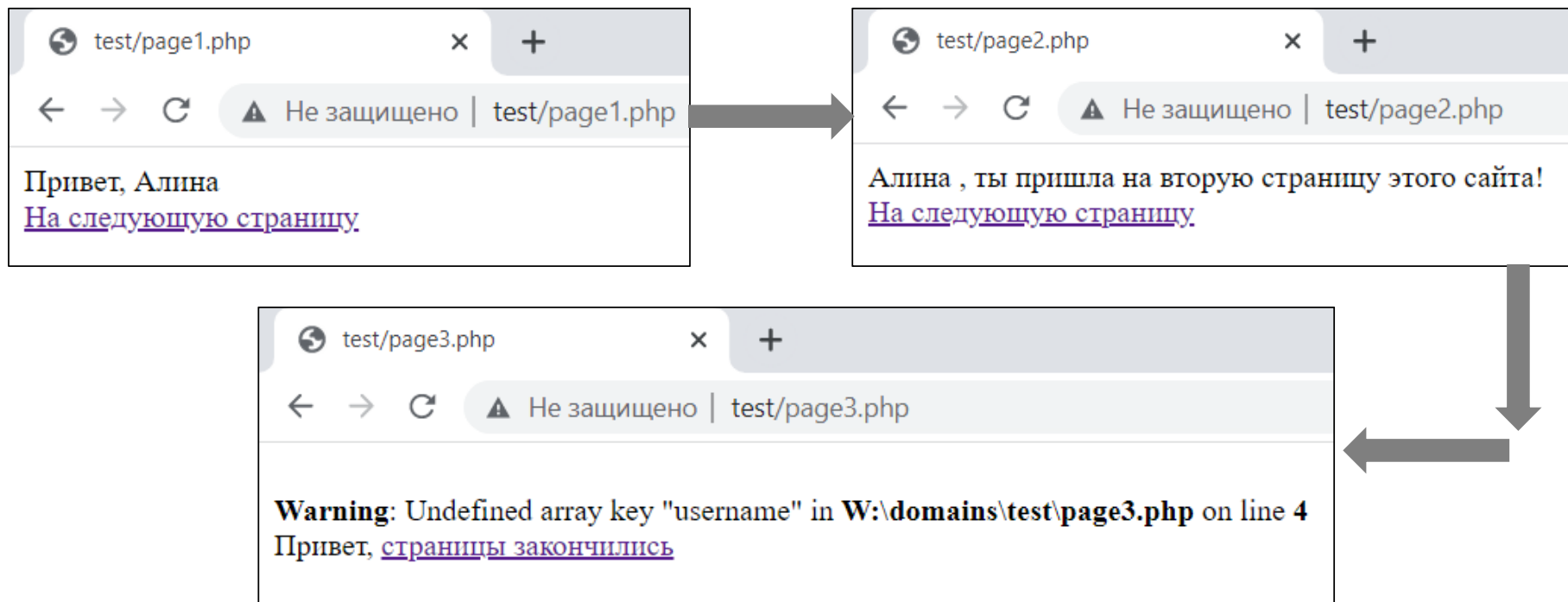
```
//page1.php
<?php
    session_start();
    $_SESSION['username'] = "Алина";
    echo 'Привет, ' . $_SESSION['username'] . "<br>";
?>
    <a href="page2.php">На следующую
    страницу </a>
```

```
//page2.php
<?php
    session_start();
    echo $_SESSION['username'] . ' , ты
    пришла на вторую страницу этого сайта!';
    echo("<br>");
?>
    <a href="page3.php">На следующую
    страницу </a>
```

```
<?php //page3.php
    session_start();
    unset($_SESSION['username']); // разрегистрировали переменную
    echo 'Привет, ' . $_SESSION['username'] . ' ';
    /* теперь имя пользователя уже не выводится */
    session_destroy(); // разрушаем сессию
?>
    <a href="page1.php">страницы закончились </a>
```

Сессии в PHP

При посещении пользователем первой страницы открывается сессия и регистрируется переменная `$username`. После этого, пользователь Алина нажимает на ссылку и попадает на страницу `page2.php`. При нажатии на ссылку, пользователь попадает на страницу `page3.php`, при этом происходит разрегистрация сеансовой переменной и уничтожение сессии.



Cookies в PHP

Cookies представляет собой специальный механизм, посредством которого сервер может разместить какие-либо данные на клиенте. При следующем обращении к серверу браузер автоматически передаст ему соответствующие cookies. Этот механизм используется для "запоминания" пользователей на сайтах, для хранения текущего состояния сайта для конкретного пользователя и некоторых других задач. Cookies могут использоваться не только серверными, но и клиентскими скриптами (например, написанными на JavaScript).

Физически cookie представляет собой обычный текстовый файл, содержащий данные в структурированном виде. Эти файлы обычно располагаются в каталоге браузера или каталогах для хранения временных файлов Интернет. В каждом браузере есть средство просмотра и управления cookies.

Cookies в PHP

В PHP создать и отправить cookie позволяют функции `setcookie()` и `setrawcookie()`. Обе функции идентичны по своему назначению, но параметры функции `setcookie()` предварительно автоматически кодируются в URL-формат.

```
<?php
    setcookie($name, $value, $expire, $path, $domain, $secure, $httponly);
?>
```

Функция `setcookie()` должна быть вызвана до того, как первые данные были отправлены в браузер (например, до первой команды `echo` или `print`). В противном случае cookie установлен не будет, а сервер выдаст предупреждение или ошибку.

Cookies в PHP

Рассмотрим подробнее каждый из параметров:

name: имя cookie, которое будет использоваться для доступа к его значению

value: значение или содержимое cookie - любой алфавитно-цифровой текст не более 4 кБайт

expire (необязательный параметр): срок действия в секундах, после которого cookie уничтожаются.

Если данный параметр не установлен или равен 0, то уничтожение cookie происходит после закрытия браузера. **path** (необязательный параметр): путь к каталогу на сервере, для которого будут доступны cookie. Если задать "/", cookie будут доступны для всего сайта. Если задать, например, "/mydir/", cookie будут доступны только из каталога /mydir/ и всех его подкаталогов. По умолчанию значением является текущий каталог, в котором устанавливаются cookie.

domain (необязательный параметр): задает домен, для которого будут доступны cookie.

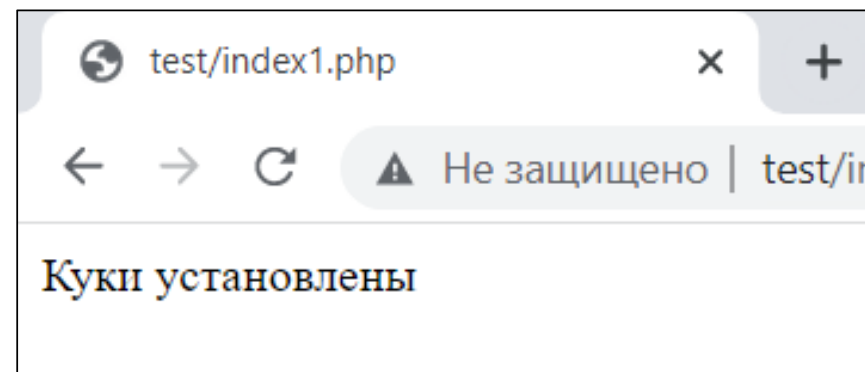
secure (необязательный параметр): указывает на то, что значение cookie должно передаваться по протоколу HTTPS. Если задано true, cookie от клиента будет передано на сервер, только если установлено защищенное соединение. По умолчанию равно false.

httponly (необязательный параметр): если равно true, cookie будут доступны только через http протокол. То есть cookie в этом случае не будут доступны скриптовым языкам, например, JavaScript. По умолчанию параметр равен false

Cookies в PHP

Сохраним cookie:

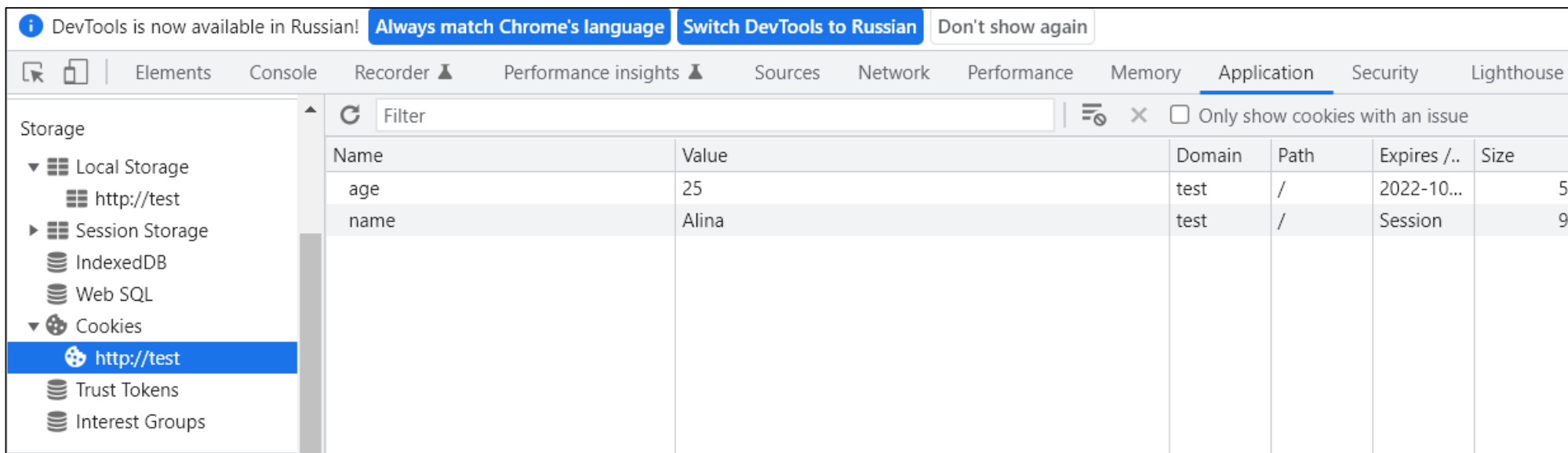
```
<?php
$name = "Tom";
$age = 36;
setcookie("name", $name);
setcookie("age", $age, time() + 3600);
// срок действия - 1 час (3600 секунд)
echo "Куки установлены";
?>
```



Здесь устанавливаются две cookie-переменные: "name" и "age". Первая cookie-переменная уничтожается после закрытия браузера, а вторая - через 3600 секунд, то есть через час.

Cookies в PHP

При необходимости мы можем увидеть сохраненные куки в браузере с помощью инструментов разработчика. Например, вид куки в Google Chrome:



The screenshot shows the Google Chrome DevTools interface with the 'Application' tab selected. In the left sidebar, the 'Storage' section is expanded, and 'Cookies' for the domain 'http://test' is selected. The main panel displays a table of cookies. The table has columns for Name, Value, Domain, Path, Expires /.., and Size. Two cookies are listed: 'age' with value '25' and 'name' with value 'Alina'. Both cookies have a domain of 'test' and a path of '/'. The 'age' cookie expires on '2022-10...' and has a size of 5. The 'name' cookie is a 'Session' cookie and has a size of 9.

Name	Value	Domain	Path	Expires /..	Size
age	25	test	/	2022-10...	5
name	Alina	test	/	Session	9

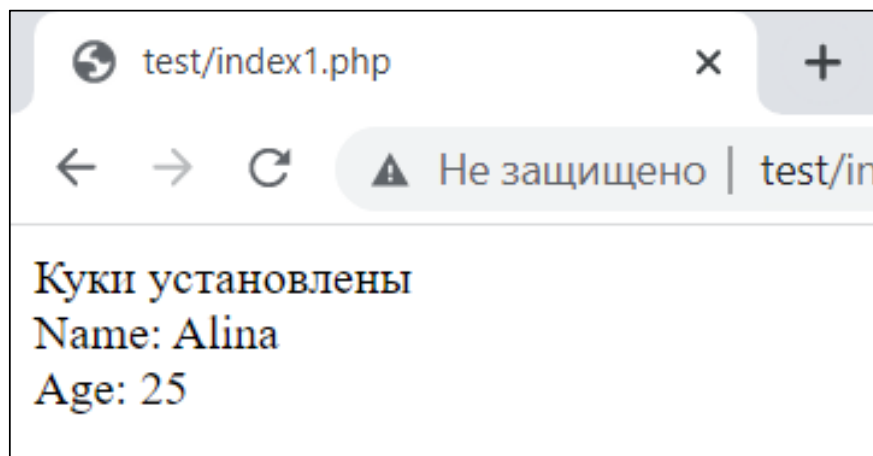
В cookie можно сохранить любую информацию, но не стоит сохранять важные с точки зрения безопасности данные, например, пароли. А если и сохранять какую-то важную информацию, то следует хранить ее в зашифрованном виде.

Cookies в PHP

Получение Cookies в PHP

Чтобы получить cookie, можно использовать глобальный ассоциативный массив `$_COOKIE`, например, `$_COOKIE["name"]`. Так, получим ранее сохраненные куки:

```
<?php
if (isset($_COOKIE["name"])) echo "Name: " . $_COOKIE["name"] . "<br>";
if (isset($_COOKIE["age"])) echo "Age: " . $_COOKIE["age"] . "<br>";
?>
```

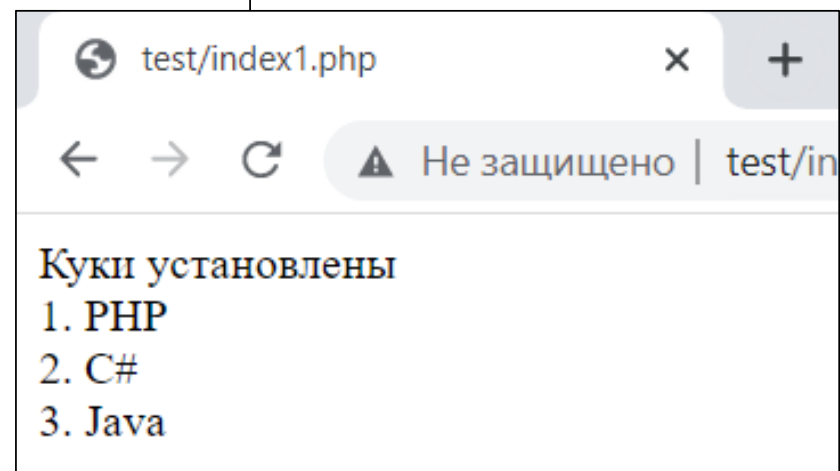


Cookies в PHP

В Cookies можно сохранять массивы

```
<?php
$name = "Tom";
$age = 36;
setcookie("lang[1]", "PHP");
setcookie("lang[2]", "C#");
setcookie("lang[3]", "Java");
echo "Куки установлены";

if (isset($_COOKIE["lang"])) {
    foreach ($_COOKIE["lang"] as $name => $value) {
        $name = htmlspecialchars($name);
        $value = htmlspecialchars($value);
        echo "$name. $value <br />";
    }
}
?>
```



Cookies в PHP

Удаление Cookies

Для удаления cookie достаточно в качестве срока действия указать какое-либо время в прошлом

```
setcookie ("name", "", time() - 3600);
```