Internet-технологии

ЛЕКЦИЯ №10
ЯЗЫК PHP

Содержание

- Работа с файлами.
- Кукисы.
- Сессии.
- Файл дополнительных настроек конфигурации .htaccess.

Работа с файлами

Работа с файлами разделяется на 3 этапа: **открытие** файла, манипуляции с **данными**, **закрытие** файла. Чтобы открыть файл используется функция fopen().

```
$fp = fopen('counter.txt', 'r');
```

Обязательными параметрами этой функции является **имя файла** и **режим** работы с файлом.

Выделяют следующие режимы работы с файлом:

r – открытие файла только для чтения.

r+ – открытие файла одновременно на чтение и запись.

w – создание нового пустого файла.

w+ – аналогичен r+, только если на момент вызова такой файл существует, его содержимое удаляется.

a – открывает существующий файл в режиме записи (на конец файла).

a+ – открывает файл в режиме чтения и записи. Содержимое файла не удаляется.

Работа с файлами

При успешном выполнении функция **fopen()** возвращает дескриптор файла, который предназначен для ссылок на файл при последующих вызовах функций для работы с файлами. При неудаче возвращается **false**.

По завершении работы с файлом необходимо его закрыть при помощи функции **fclose()**, которая в качестве аргумента принимает дескриптор закрываемого файла. В случае успеха возвращается true, в случае неудачи – false.

```
$file_hendler = fopen("report", "a");
   //or exit("Невозможно открыть файл report");
if (!$file_hendler) {
  echo("Невозможно открыть файл"); //die()
}
else {
  // функции работы с файлами
}
```

Работа с файлами

fgetc — Считывает символ из файла;
fgets — Читает строку из файла;
fgetss — Прочитать строку и отбросить HTML-теги;
file_get_contents — Читает содержимое файла в строку;
file_put_contents — Пишет строку в файл;
file — Читает содержимое файла и помещает его в массив;
readfile — используется при выводе файла пользователю;
fileatime — Возвращает время последнего доступа к файлу;
filemtime — Возвращает время последнего изменения файла;
filesize — Возвращает размер файла;
filetype — Возвращает тип файла;
fstat — Получает информацию о файле используя открытый файловый указатель;
stat — Возвращает информацию о файле.

```
$fp = fopen('data.txt', 'w');
    fwrite($fp, '1');
    fwrite($fp, '23');
    fclose($fp);
```

Работа с файлами. Пример

```
$filename = "lb4_9_counter.dat";
$fp = @fopen($filename, "r");
if($fp)  {
   $counter = fgets($fp,10);
   fclose($fp);
}
else  {
    $counter = 0;
}
$counter ++;
echo $counter;
```

```
$fp = @fopen($filename, "w");
if($fp)  {
    $counter = fputs($fp, $counter);
    fclose($fp);
}
```

Функция **file_put_contents** идентична последовательным успешным вызовам функций fopen(), fwrite() и fclose().

В случае, если оператор подавления @ предшествует какому-либо выражению в коде PHP-скрипта, любые сообщения об ошибках, генерируемые этим выражением, будут проигнорированы.

Использование функции error_reporting

Функция **error_reporting()** задает значение директивы error_reporting во время выполнения. Используя эту функцию, можно задать уровень ошибок времени выполнения скрипта, которые попадут в отчет. Если необязательный аргумент level не задан (в виде битовой маски или **именованных констант**), error_reporting() вернет текущее значение уровня протоколирования ошибок.

Примеры именованых констант, определяющих вывод ошибок:

E_ALL – все поддерживаемые ошибки и предупреждения;

E_ERROR – фатальные ошибки времени выполнения;

E_NOTICE – уведомления времени выполнения;

E_WARNING – предупреждения времени выполнения скрипта (нефатальные ошибки);

E_CORE_WARNING – предупреждения (нефатальные ошибки), которые происходят во время начального запуска PHP;

E_DEPRECATED – уведомления времени выполнения об использовании устаревших конструкций.

Использование функции error_reporting

```
// Выключение протоколирования ошибок
error_reporting(0);

// Добавлять сообщения обо всех ошибках, кроме E_NOTICE
error_reporting(E_ALL & ~E_NOTICE);

// Добавлять в отчет все PHP ошибки
error_reporting(E_ALL);
error_reporting(-1);
ini_set('error_reporting', E_ALL);
```

Включение внешних файлов

Выражение include включает и выполняет указанный файл в текущем PHP-скрипте. Этой инструкцией языка удобно пользоваться при наличии одинаковых фрагментов кода во многих PHP-скриптах.

```
include "file.txt";
include 'file.php';
include 'http://www.example.com/file.php?foo=1&bar=2';
```

Конструкция include выдаст предупреждение, если не сможет найти файл; поведение отлично от аналогичной конструкции require, которая выдаст ошибку.

Выражение include_once включает и выполняет указанный файл во время выполнения скрипта. Если код из файла уже один раз был включен, он не будет включен и выполнен повторно.

Включение внешних файлов. Пример

Test2.php:

```
<!DOCTYPE HTML>
<html>
<head>
 <title>Тест</title>
 <meta charset="utf-8">
</head>
<body>
<?php
  echo "A $color $fruit"; // Undefined variable
  include 'vars.php';
  echo "A $color $fruit"; // A green apple
?>
</body>
</html>
```

Когда файл включается, его код наследует ту же область видимости переменных, что и строка, на которой произошло включение. Все переменные, доступные на этой строке во включающем файле будут также доступны во включаемом файле.

vars.php:

```
<?php
 $color = 'green';
 $fruit = 'apple';
?>
```

Включение внешних файлов. Пример

Включаемые файлы могут возвращать значения подобно функциям. Использование оператора **return** прерывает выполнение этого файла так же, как и функции.

Test2.php:

```
<?php
  $res = include("vars.php");
  echo "Включаемый файл вернул $res";
?>
```

vars.php:

```
<?php
  $a = 7 * 8;
  return $a;
?>
```

Инструкцию include() можно использовать **внутри цикла** или включать в тело **условного оператора**. В цикле include() выполняется при каждой итерации. Это можно использовать для включения нескольких файлов.

```
for ($i=1; $i<=5; $i++)
    include("incfile{$i}.htm");
```

Cookies

Cookies – это механизм хранения данных в браузере для идентификации возвращающихся посетителей и хранения параметров веб-страниц (например, переменных) в виде пар **ключ-значение**. Файлы Cookies представляют собой обыкновенные **текстовые файлы**, которые хранятся на диске **у посетителей** сайтов. Файлы Cookies и содержат ту информацию, которая была в них записана сервером.

Применяется обычно для:

- аутентификации пользователя;
- хранения персональных предпочтений и настроек пользователя;
- отслеживания состояния сеанса доступа пользователя;
- ведения статистики о пользователях.

Данные, хранящиеся в cookie, доступны в суперглобальных массивах, таких как $_COOKIE, (устаревший – $HTTP_COOKIE_VARS), а также в $_REQUEST.

```
echo 'Привет, ' . htmlspecialchars($_COOKIE["name"]) . '!';
```

Cookies. Функция setcookie()

Чтобы установить cookies, которое будет передано клиенту, используется функция **setcookie()**. Cookies являются частью HTTP-заголовка, поэтому, как и любой другой заголовок, cookie должны передаваться до того, как будут выведены какие-либо другие данные скрипта.

Для этой функции можно указать шесть параметров:

- **name** – задает имя, закрепленное за Cookie;
- **value** – определяет значение переменной (строка);
- **expire** – время "жизни" переменной (целое число). Если данный параметр не указать, то Cookie будут "жить" до конца сессии, то есть до закрытия браузера.
- path – путь к Cookie (строка) и domain – домен (строка). Служат для ограничения доступа к кукисам в пределах заданного домена и каталога;
- secure – передача Cookie через защищенное HTTPS-соединение;
- **httponly** – если задано TRUE, cookie будут доступны только через HTTP протокол.

Cookies. Функция setcookie()

Пример установки Cookies:

```
<?php
    SetCookie("Test","Value");       // устанавливаем Cookie до конца сессии
    SetCookie("My_Cookie","Value", time()+3600);         // на один час
    setcookie("name", $value, "/web/index.php", ".server.com");
?>
```

Если необходимо присвоить множество значений одной переменной cookie, их можно присвоить как массив. Например:

```
<?php
    setcookie("MyCookie[foo]", "Тест 1", time()+3600);
    setcookie("MyCookie[bar]", "Тест 2", time()+3600);
?>
```

Cookies. Чтение и удаление

```php
<?php
   echo $_COOKIE["TestCookie"];
   echo $HTTP_COOKIE_VARS["TestCookie"];
   print_r($_COOKIE);

   if (isset($_COOKIE['MyCookie'])) {
      foreach ($_COOKIE['MyCookie'] as $name => $value) {
         $name = htmlspecialchars($name);
         $value = htmlspecialchars($value);
        echo "$name : $value <br />\n";
      }
   }
    // установка даты истечения срока действия на час назад
    setcookie ("TestCookie", "", time() - 3600);
?>
```

```
foo : Тест 1
bar : Тест 2
```

Cookies. JavaScript

В JavaScript куки доступны с помощью свойства **cookie** объекта **document**. Создать cookie можно следующим образом:

```
document.cookie = "name=значение; expires=дата; path=путь; domain=домен; secure";
document.cookie = "username=Вася; expires=15/02/2011 00:00:00";
```

Получение cookie:

```
var x = document.cookie;
```

```
login=Вася;                                    other.php:8
hashpass=0123456789abcdtf;
PHPSESSID=anf50p7gv1o3fjnasq50j0k8v0
```

Пример:

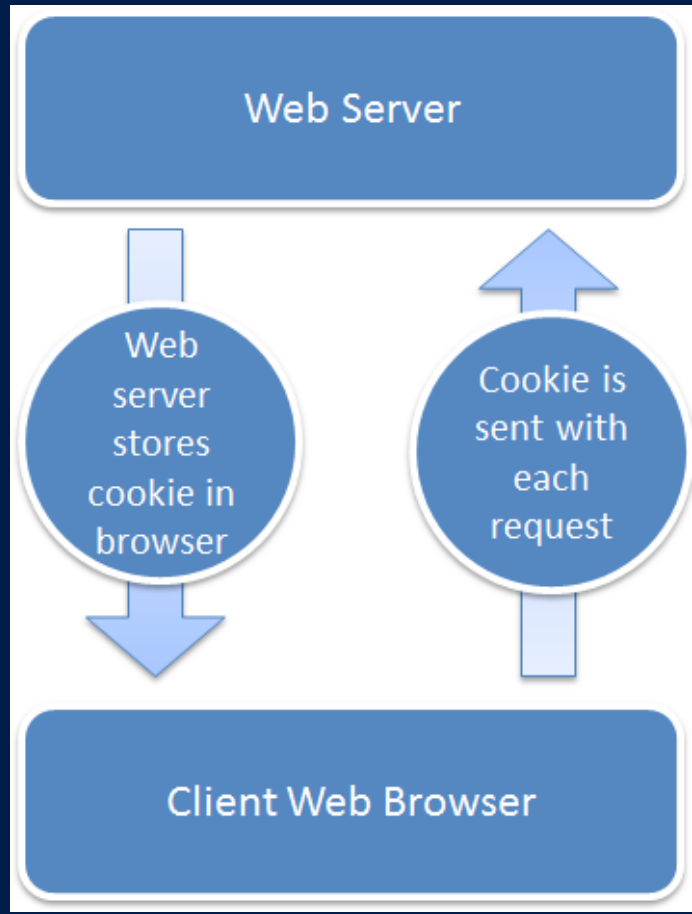
```
<script>
  document.cookie = "login=Вася; expires=15/02/2011 00:00:00";
  document.cookie = "hashpass=0123456789abcdtf; expires=15/02/2011 00:00:00";
  console.log(document.cookie);
</script>
```

Ограничения HTTP Cookies

На протяжении долгого времени HTTP Cookies были единственным кроссбраузерным способом сохранить данные, которые будут доступны после перезагрузки страницы.

Однако у cookie есть важные особенности, например:

- cookie имеет **ограничение по размеру**, 4-10 килобайт данных;
- cookie участвуют в формировании каждого http-запроса к серверу, т.е. при каждом запросе все cookie автоматически отправляются вместе с запросом, что **увеличивает трафик**;
- cookie сопоставлены с web-сайтом и, если пользователь работает с сайтом через две вкладки, он оперирует **одними и теми же данными cookie**. Этот нюанс может нарушить правильную работу сайта и ограничивает применение cookie.



Сессии

Сессии – это механизм, который позволяет создавать и использовать переменные, сохраняющие свое значение в течение всего времени работы пользователя с сайтом. Для каждого пользователя имеют различные значения и могут использоваться на любой странице сайта до выхода пользователя из системы.

Задача идентификации пользователя решается путем присвоения каждому пользователю уникального идентификатора сессии (SID, Session IDentifier). Он генерируется PHP в тот момент, когда пользователь заходит на сайт, и уничтожается, когда пользователь уходит с сайта, и представляет собой строку из 32 символов (например, ac4f4a45bdc893434c95dcaffb1c1811).

При работе с сессиями различают следующие этапы:

- открытие сессии;
- регистрация переменных сессии и их использование;
- закрытие сессии.

Сессии. Регистрация переменных

```php
<?php
 // Инициируем сессию
 session_start();

 // Помещаем значение в сессию
 $_SESSION['name'] = "value";
 // Помещаем массив в сессию
 $arr = array("first", "second", "third");
 $_SESSION['arr'] = $arr;

 // Выводим ссылку на другую страницу
 echo "<a href='other.php'>Другая страница</a>";

 //session_destroy();
?>
```

```php
<?php
 // Инициируем сессию
 session_start();

 echo "<pre>";
 print_r($_SESSION);
 echo "</pre>";
?>
```

```
Array
(
    [name] => value
    [arr] => Array
        (
            [0] => first
            [1] => second
            [2] => third
        )

)
```

Файл .htaccess

.htaccess (от. англ. hypertext access) – файл дополнительной конфигурации веб-сервера, который дает возможность задавать большое количество дополнительных параметров, не предоставляя доступа к главному конфигурационному файлу.

Позволяет настраивать перенаправление, управляемый доступ к каталогам, переназначение типов файлов и т.д.

Вы можете использовать htaccess с разными настройками для разных каталогов. Действие опций htaccess распространяет сверху вниз по дереву каталогов до самой глубокой вложенности, пока не будут отменены другим htaccess.

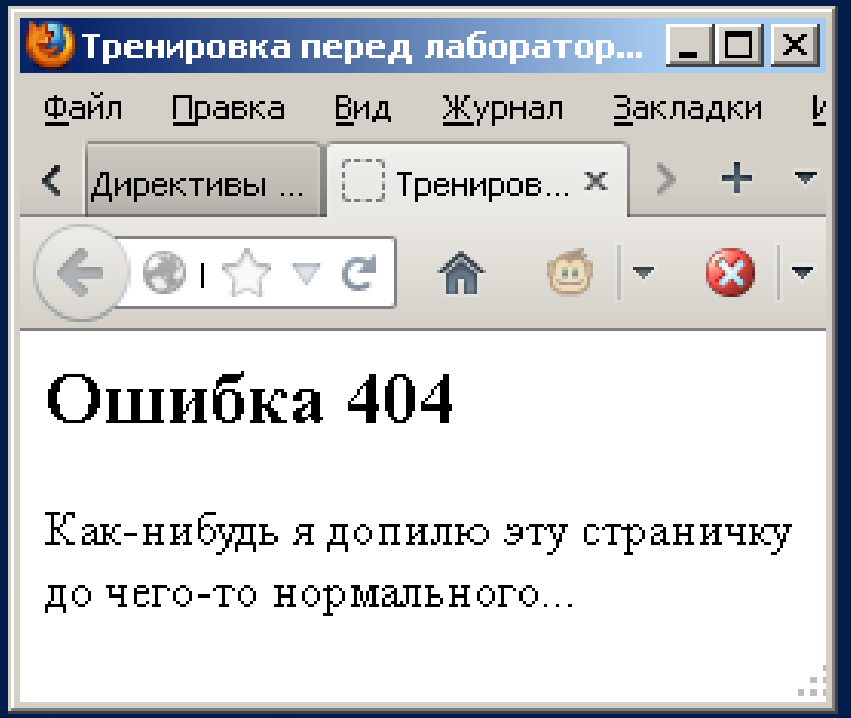
Необязательно полностью перечислять все опции в дочерних .htaccess если они не изменяются. Достаточно указать (переназначить) только те опции и директивы, которые изменяются. Остальные опции также унаследуются от родителя.

```
DirectoryIndex index.html index.php
```

.htaccess. Пример перенаправления страницы 404

```
# Not found
ErrorDocument 404 /it1lab4/index404.html
# Authorization Required
ErrorDocument 403 /it1lab4/403.html
# Internal Server Error
ErrorDocument 500 /it1lab4/500.html
```

При этом размер конечного файла должен быть больше 515 байт, иначе некоторые браузеры могут подменить страницу своей стандартной.



Вопросы

- Порядок работы с файлами в языке PHP. Функция fopen и ее параметры.
- Как настроить уровень вывода сообщений об ошибках? Какие уровни вам известны?
- Для чего применяется конструкция include? В чем отличия инструкции include_once?
- Что такое Cookies?
- Для чего используется функция setcookie()? Каковы ее параметры?
- Возможен ли доступ к значениям Cookies на стороне клиента?
- Недостатки HTTP Cookies.
- Для чего используются сессии?
- Как установить и удалить значения переменных сессии?
- Назначение файла .htaccess.
- Для чего используется # при настройке файла .htaccess?