

Оглавление

Лекция 1. Введение в PHP.....	2
ЛЕКЦИЯ 2. ДВУМЕРНЫЕ МАССИВЫ, ОПЕРАТОРЫ, ФУНКЦИИ.	7
ЛЕКЦИЯ 3. РЕКУРСИЯ.	14
Лекция 4. Формы.....	16
Лекция 5. СУБД.....	19
Лекция 6. Введение в MySQL.	22
Связь таблиц.	25
PHP + MySQL	28
Лекция 7.	30
Теперь рассмотрим рекурсивный вывод таблицы.....	31
Лекция 8. Session and Cookie.....	34
Session. Пример работы.	35
Cookie. Пример работы.	36

Лекция 1. Введение в PHP.

PHP (Personal Home Page Tools – инструменты для создания персональных веб-страниц) — скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений.

Синтаксис PHP похож на синтаксис языка Си. Некоторые элементы, такие как ассоциативные массивы (позволяет хранить пары вида «(ключ, значение)») и цикл `foreach`, заимствованы из Perl.

Для написания простейшего скрипта не требуется описывать какие-либо переменные, используемые модули и т. п. Любой скрипт может начинаться непосредственно с оператора PHP.

- Справочники в интернете: php.net(английская версия), php.ru (русская)
- При создании файла важно указывать разрешение `.php`, иначе ничего не выполнится.

HW.php Самая простая программа вывода текста на экран.

```
<?php  
echo 'Hello, world!';  
?>
```

Также возможен более короткий вариант вывода строки:

```
<?= 'Hello, world!' ?>
```

Открывающий тег вида `<?=
используемых для вывода строки. PHP исполняет код, находящийся внутри ограничителей, таких как <?php ?>. Всё, что находится вне ограничителей, выводится без изменений. В основном это используется для вставки PHP-кода в HTML-документ, например, так:`

TestPhp.php

```
<html>
<head><title>Тестируем PHP</title></head>
<body><?php
    echo "Привет мир!";
?>
</body>
</html>
```

Помимо ограничителей `<?php ?>`, допускается использование сокращённого варианта `<? ?>`.

Имена переменных начинаются с символа `$`, тип переменной объявлять не нужно. Имена переменных и констант чувствительны к регистру символов. Имена классов, методов классов и функций к регистру символов не чувствительны. Переменные обрабатываются в строках, заключённых в двойные кавычки, и heredoc-строках (строках, созданных при помощи оператора `<<<`). Переменные в строках, заключённых в одинарные кавычки, не обрабатываются.

TestVal.php

```
<?php
$a = 1; # a получает тип данных int
$b = 1.0; // b получает тип данных double
$c = "1"; # c получает тип данных string
echo $a, $b, $c;
/* функция echo выведет на экран
значения переменных a, b, c */
?>
```

При запуске данного скрипта мы увидим: 111.

PHP рассматривает переход на новую строку как пробел, так же как HTML и другие языки со свободным форматом. Инструкции разделяются с помощью точки с запятой (;), за исключением некоторых случаев, после объявления конструкции if/else и циклов.

Переменные в функцию можно передавать как по значению, так и по ссылке (используется знак &).

PHP поддерживает три типа комментариев:

- в стиле языка Си (ограниченные /* */);
- C++ (начинающиеся с // и идущие до конца строки);
- оболочки UNIX (с # до конца строки).

Приведение типов автоматическое:

TestType.php

```
<?php
$a = 1;
$b = (double)$a; //получаем вид 1.0
$c = (string)$a; //теперь наше число стало строкой "1";
?>
```

Пример объявления одномерного массива:

EchoArr.php

```
<?php
$a[0] = "t"; $a[1] = "e"; $a[2] = "s"; $a[3] = "t";
for( $i = 0; $i < 4; $i++){
    echo $a[i]; // на выходе получим: test
}
?>
```

Пример цикла:

TestCycle.php

```
<?php
    for( $i = 0; $i < 345; $i++)
    {
        echo $i."\\n"; // на выходе получим столбик из чисел, идущих по порядку
    }
?>
```

Задача: вывести на экран сумму чисел от 1 до 10

EchoSumEl.php

```
<?php
    for( $i = 0; $i < 11; $i++)
    {
        $sum = $sum + $a[$i];
    }
    echo $sum;
?>
```

Задача: вывести на экран сумму элементов массива, состоящего из пяти элементов, каждый из которых равен 5

EchoSumArr.php

```
<?php
    $a[0] = 5; $a[1] = 5; $a[2] = 5; $a[3] = 5; $a[4] = 5;
    for( $i = 0; $i < 345; $i++)
    {
        $sum = $sum + $a[$i];
    } echo $sum;
?>
```

Задача: сложить 2 вектора

SumVect.php

```
<?php
$a[0] = 1; $a[1] = 2; $a[2] = 3;
$b[0] = 0; $b[1] = 1; $a[2] = 2;
$c[0] = 0; $c[1] = 0; $a[2] = 0;
for( $i = 0; $i < 3; $i++)
{
    $c[$i] = $a[$i] + $b[$i];
    echo $c[$i];
}
?>
```

Замечание: При написании ограничителей таким образом `<? php ?>` (пробел перед php) может появляться ошибка, т.к. php воспринимается как неправильно объявленная переменная.

ЛЕКЦИЯ 2. ДВУМЕРНЫЕ МАССИВЫ, ОПЕРАТОРЫ, ФУНКЦИИ.

Пример двумерного массива:

Test2DimArr.php

```
<?php
$a[0][0] = 5; $a[0][1] = 3; $a[0][2] = 9;
$a[1][0] = 7; $a[1][1] = 1; $a[1][2] = 4;
$a[2][0] = 2; $a[2][1] = 8; $a[2][2] = 0;
$a[3][0] = 6; $a[3][1] = 4; $a[3][2] = 1;
?>
```

$a[3][2] = 1$ — в строке 3 во втором столбце лежит значение 1;

Условные операторы: If/else

TestIfElse.php

```
<?php
$a = 1; $b = 2;
if( $a > $b)
{
    echo "не верно!";
}
else if($a == $b)
{
    echo "равенство не выполняется";
}
else
{
    echo "верно";
}
?>
```

TestSwitch.php

```
<?php
switch($переменная){
    case "значение1": оператор1;
        break;
    case "значение2": оператор2;
        break;
    case "значение3": оператор3;
        break;
    [default: оператор4;
        break;]
}?>
```

Операторы

- Операторы сравнения: `>`(больше) `<`(меньше) `!=`(не равно) `==`(равно)
- Побитовые операторы: `&`(и) `|`(или)
- Логические операторы: `and`(и) `or`(или) `xor`(исключающее или) `!`(отрицание) `&&`(и) `||`(или)

Функции

Функции можно объявлять как в начале, так и в конце; их можно передавать как переменные; `function` — директива

TestFunc.php

```
<?php
function sum($a, $b)
{
    $c = $a + $b; // a, b, c локальные переменные
    return $c;
}
```



```
$a = 10; $y = 15; $c = sum($a, $b);  
?>
```

Задача: вывести массив \$A на экран.

EchoMatrix.php

```
<?php  
function echo_matrix($A)  
{  
    for($i = 0; $i < sizeof($A); $i++)  
    {  
        for(($j = 0; $j < sizeof($A); $j++)  
        {  
            echo $A[$i][$j];  
        }  
        echo "</br>";  
    }  
}  
echo_matrix($A);  
?>
```

Замечание: Чтобы выполнить поставленную задачу, конечно же нужно задать вначале массив \$A перед вызовом самой функции.

Вывод таблицы в тегах HTML:

Table.html

```
<table>(или <table border ="1"> - эта строка добавляет толщину ограничивающих черт  
таблицы)  
<tr>(обозначаем начало строки)<td>(обозначаем начало  
столбца)00</td><td>01</td><td>02</td></tr>  
<tr><td>10</td><td>11</td><td>12</td></tr>  
<tr><td>20</td><td>21</td><td>22</td></tr>
```

</table>

Вывод матрицы в таблице:

EchoMatrix.php

```
<?php
function echo_matrix($A)
{
    echo "<table border = \'1\'>";
    for($i = 0; $i < sizeof($A); $i++)
    {
        echo "<tr>";
        for(($j = 0; $j < sizeof($A); $j++)
        {
            echo "<td>".$A[$i][$j]."</td>";
        }
        echo "</tr>";
    }
    echo "</tr>";

}
echo_matrix($A);
?>
```

Скалярное произведение двух векторов:

Scalyar.php

```
<?php
$a[0] = 1; $a[1] = 10; $a[2] = 5;
$b[0] = 3; $b[1] = 2; $b[2] = 1;
$c = 0;
for( $i = 0; $i < sizeof($a); $i++)
{
```

```
$c += $a[$i]*$b[i];  
}  
?>
```

Решение этой же задачи с помощью функции:

Scalyar2.php

```
<?php  
$a[0] = 1; $a[1] = 10; $a[2] = 5;  
$b[0] = 3; $b[1] = 2; $b[2] = 1;  
$c = 0;  
function scalyar ($a, $b){  
    for( $i = 0; $i < sizeof($a); $i++){  
        $c += $a[$i]*$b[i];  
    }  
    return $c;  
}  
echo $c = scalyar($a, $b);  
?>
```

В реальных проектах обычно используют много файлов, в которых лежат разные части кода. А для того, чтобы одну и ту же функцию не нужно было вызывать несколько раз, можно создать свою «библиотеку», которую можно подключить в другой файл при помощи include.

Include.php

```
<?php  
Include 'lib.php';  
$c = 0; $a = "1"; $b = "10"; $z = "";  
$z = concat($c, $a, $b);  
echo concat($c, $a, $b);
```

?>

В качестве дополнительного упражнения можно написать функцию сложения двух матриц.

Задание: написать функцию умножение матрицы на вектор

MultipleMatrVect.php

```
<?php
$b[0] = 1; $b[1] = 1; $b[2] = 1;

$a[0][0] = 1; $a[0][1] = 1; $a[0][2] = 1;
$a[1][0] = 1; $a[1][1] = 1; $a[1][2] = 1;
$a[2][0] = 1; $a[2][1] = 1; $a[2][2] = 1;
$c[0] = 0; $c[1] = 0; $c[2] = 0;
function multipleMandV ($a, $b, $c){
    for( $i = 0; $i < sizeof($a); $i++){
        $c += $a[$i]*$b[i];
    }
    return $c;
}
echo $c = multipleMandV ($a, $b, $c);
?>
```

Задание: написать функцию умножения матрицы на матрицу

MultipleMatrMatr.php

```
<?php
$a[0][0] = 1; $a[0][1] = 1; $a[0][2] = 1;
$a[1][0] = 1; $a[1][1] = 1; $a[1][2] = 1;
$a[2][0] = 1; $a[2][1] = 1; $a[2][2] = 1;
$b[0][0] = 1; $b[0][1] = 1; $b[0][2] = 1;
$b[1][0] = 1; $b[1][1] = 1; $b[1][2] = 1;
$b[2][0] = 1; $b[2][1] = 1; $b[2][2] = 1;
```

```

$C[0][0] = 0; $C[0][1] = 0; $C[0][2] = 0;
$C[1][0] = 0; $C[1][1] = 0; $C[1][2] = 0;
$C[2][0] = 0; $C[2][1] = 0; $C[2][2] = 0;

function multipleMandM ($A, $B, $C){
    $contant = "<table border = \"1\">";
    for( $i = 0; $i < sizeof($C); $i++){
        $contant.= "<tr>";
        for ($j= 0; $j < sizeof($C); $j++){
            for($k = 0; $k < sizeof($C); $k++){
                $C[$i][$j] += $A[$i][$k]*$B[$k][$j];
            }
            $contant.="<td>".$C[$i][$j]."</td>";
        }
        $contant.= "</tr>";
    }
    return $c;
    $contant.= "</table>";
    Return $contant;
}
echo $C = multipleMandM ($A, $B, $C);
?>

```

Вывод вектора.

EchoVect.php

```

<?php function echoVector($a){
    for($i=0; $i<sizeof($a); $i++){
        $contrnt.= "<tr><td>".$a[$i]."</td></tr>";
    }
    $content = "</table>";
    return $content;
}

```

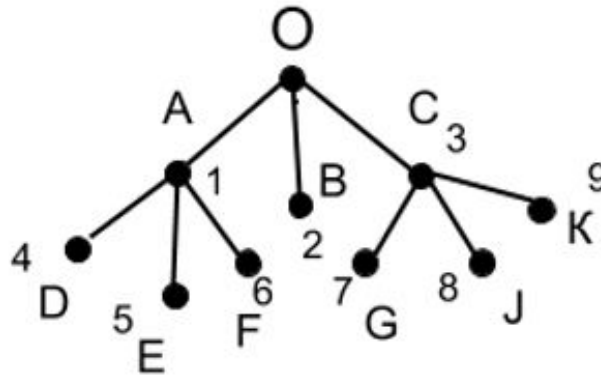
```
}?>
```

ЛЕКЦИЯ 3. РЕКУРСИЯ.

Рекурсия (вызов функции внутри этой же функции)

Дерево виде таблицы и в графической форме:

id	<u>pid</u>	<u>val</u>
1	0	A
2	0	B
3	0	C
4	1	D
5	1	E
6	1	F
7	3	G
8	3	J
9	3	K



EchoGraph.php

```
<?php
```

```
$struct[0][0] = 1; $struct[0][1] = 0; $struct[0][2] = A;  
$struct[1][0] = 2; $struct[1][1] = 0; $struct[1][2] = B;  
$struct[2][0] = 3; $struct[2][1] = 0; $struct[2][2] = C;  
$struct[3][0] = 4; $struct[3][1] = 1; $struct[3][2] = D;  
$struct[4][0] = 5; $struct[4][1] = 1; $struct[4][2] = E;  
$struct[5][0] = 6; $struct[5][1] = 1; $struct[5][2] = F;  
$struct[6][0] = 7; $struct[6][1] = 3; $struct[6][2] = G;  
$struct[7][0] = 8; $struct[7][1] = 3; $struct[7][2] = J;  
$struct[8][0] = 9; $struct[8][1] = 3; $struct[8][2] = K;
```

```
function DeepGraph($count)
```

```
{  
    for($i = 0; $i < $count; $i++)  
    {  
        echo “_”;
```

```

    }
}
function GetChildren($id, $struct, $count)
{
    for($i=0; $i<sizeof($struct); $i++)
    {
        if($struct[$i][1] == $id)
        {
            DeepGraph($count);
            echo $struct[$i][2];
            echo "</br>";
            $count +=1;
            GetChildren($struct[$i][0], $struct, $count);
            $count -=1;
        }
    }
}
$count = 0;
$id = 0;
GetChildren($id, $struct, $count);
?>

```

Для вывода графа с вычислением глубины добавили переменную, которая ее вычисляет.

Лекция 4. Формы.

Test.html

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset = "utf-8">
<title>My form</title>
</head>
<body>
<form action = "Post.php" metod = "POST">
<input type = "text" name = "val">
<input type = "submit" name = "OK" value = "OK">
</form>
```

Используем передачу данных методом POST.

Post.php

```
<?php
if(isset($_POST['val'])){
    echo $_POST['val'];
}else echo "Переменная val не задана";
?>
```

Передача параметров методом GET (все переменные и их значения передаются прямо через адрес).

Get.php

```
<?php
if(isset($_GET ['val']))
{
    echo $_GET['val'];
}
```

```

}else if(isset($_GET ['newval']))
{
    echo $_GET['newval'];
}else echo "Пока";
?>

```

test1: <http://myhost/Get.php?val=1> ← гиперссылка

test2: <http://myhost/Get.php?val=2>

test3: <http://myhost/Get.php?val=3&newval=100>

Так эти ссылки выглядят при открытии страницы браузера →

[test1](#) [test2](#) [test3](#)

Page.html

```

<html>
<head><meta charset = "utf-8">
<title>Страница с примером</title>
</head><body>
<a href = "/Get.php?val = 1">test1</a>
<a href = "/Get.php?val = 2">test2</a>
<a href = "/Get.php?val = 3&newval = 100">test3</a>
</body>

```

FormExample.php

```

<?php
echo "<a href = \"/FormExample.php?id=1\">test</a>";
echo "<form method = \"POST\" action= \"FormExample.php\">";
echo "id:<input type = \"text\" name = \"id\" size = \"2\">";
echo "<input type = \"submit\" value = \"run\">";
echo "</form>";
if(isset($_GET['$id']))
{

```

```

    echo "GET:id = ".$_GET['id'].<br>";
} else if(isset($_POST['id']))
{
    echo "POST:id = ".$_POST['id'].<tr>";
} else echo "id не передано!";
?>

```

TextField.html

```

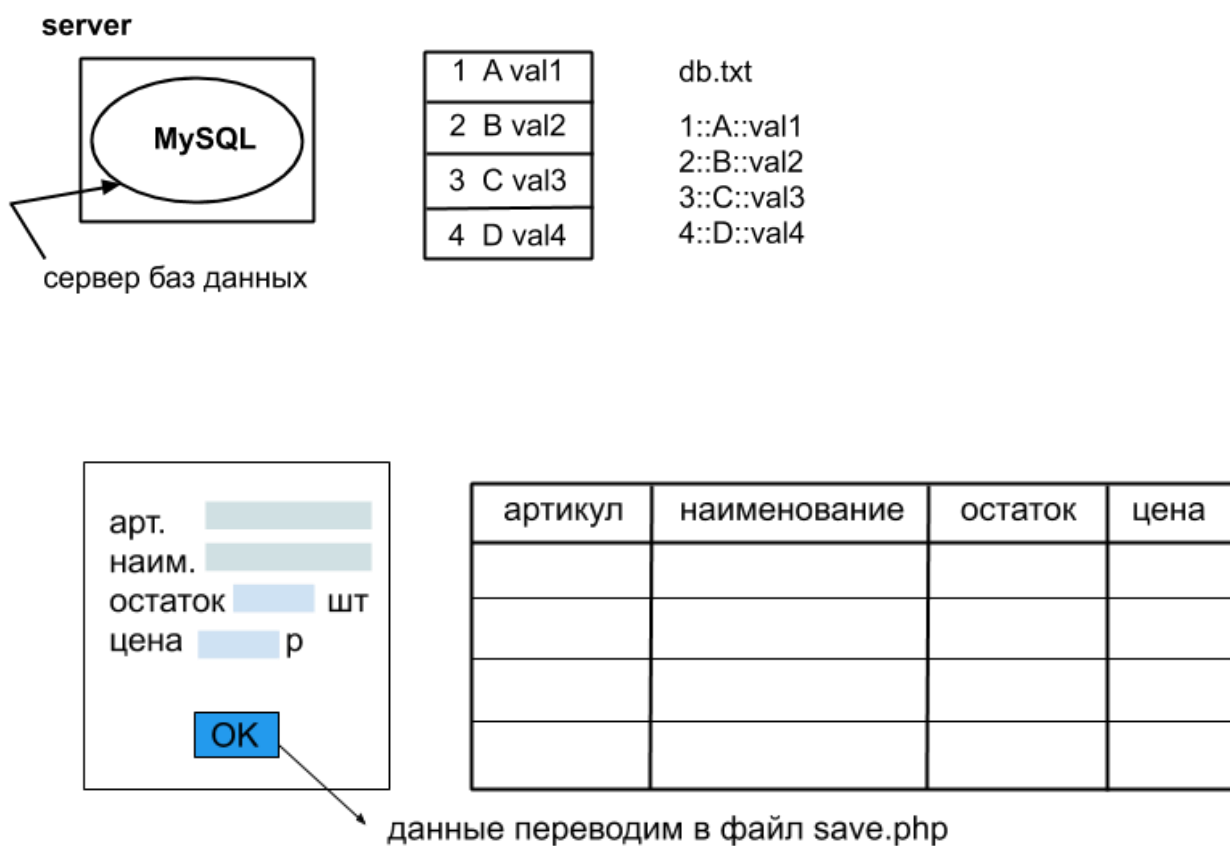
<HTML>
<HEAD><meta charset = "utf-8">
<TITLE>Text field</TITLE>
</HEAD>
<BODY>
<form>description<br>
<textarea name = "text" cols = "20" rows = "10">some default text</ textarea>
<select size = "10" name = "val">
<option value = "Test1">Test1</option>
<option value = "Test2">Test2</option>
<option value = "Test3">Test3</option>
</select>
"M<input type = "radio" name = "sex" val = "1">
"W<input type = "radio" name = "sex" val = "0">
</BODY>
</HTML>
</body>

```

Лекция 5. СУБД.

В современной технологии баз данных предполагается, что создание базы данных, её поддержка и обеспечение доступа пользователей к ней осуществляется централизованно с помощью специального программного инструментария – системы управления базами данных.

Система управления базами данных (СУБД) – это совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования БД многими пользователями. (или проще: набор таблиц, где хранятся данные. Могут иметь различные типы данных.)



Запись данных в файл:

```
<?php
if($fL1 = fopen("db.txt", "w")); //fL1 – дескриптор файл
echo "File opened";
$fL2 = fopen("./db2.txt", "w");
if($fL3 = fopen("./db3.txt", "w") );
{
    echo "File db3.txt opened! ";
```

```

} else
    echo "File error!";
fwrite(fL3, "Some text\n");
fwrite($fL3, "Another text\n");
fclose($fL3);
fclose($fL2);
fclose($fL1); // ← даже если ничего не записали файл нужно закрыть!
?>

```

fopen — Открывает файл или URL

fwrite — Бинарно-безопасная запись в файл

fclose — Закрывает открытый дескриптор файла

“r” открывает файл только для чтения; помещает указатель в начало файла.

“w” открывает файл только для записи; помещает указатель в начало файла и обрезает файл до нулевой длины. Если файл не существует – пробует его создать.

“r+” открывает файл для чтения и записи и помещает указатель в начало файла.

“w+” открывает файл для чтения и записи; помещает указатель в начало файла и обрезает файл до нулевой длины. Если файл не существует – пытается его создать.

```
fwrite($fL2, $arr[$i][0].":текст".$arr[$i][1]."\n");
```

arr		
1	A	val1
2	B	val2
3	C	val3
4	D	val4

db.txt

```

1::A::val1
2::B::val2

```

Запись данных в массив:

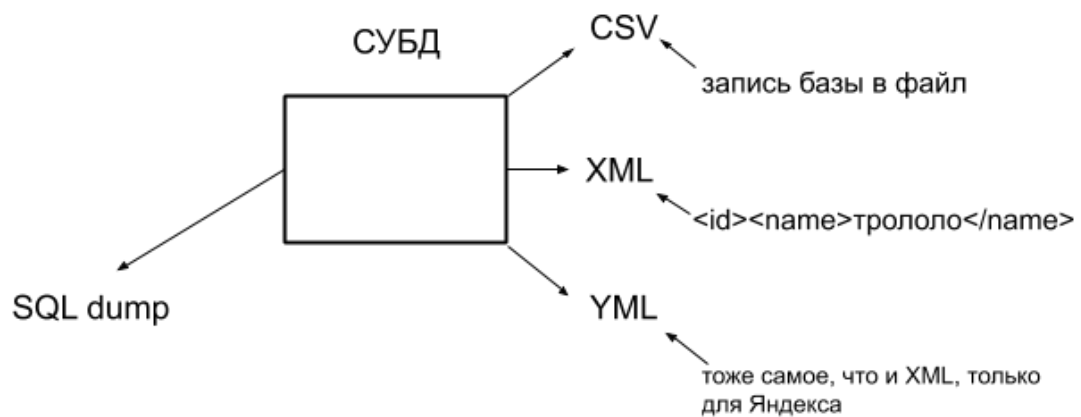
```
<?php
```

```

$db = file("./db");
for($i = 0; $i < sizeof($db); $i++)
{
    $raw = explode("::", $db[$i][0]);
    $arr[$i][0] = $raw[0];
    $arr[$i][1] = $raw[1];
    $arr[$i][2] = $raw[2];
}
// 3 последние твроки кода можно записать по-другому
for($j = 0; $j < sizeof($raw); $j++)
{
    $arr[$i][$j] = $raw[$j]
}
?>

```

Форматы хранения SQL данных



YML также можно встретить под аббревиатурой YAML на просторах интернета.

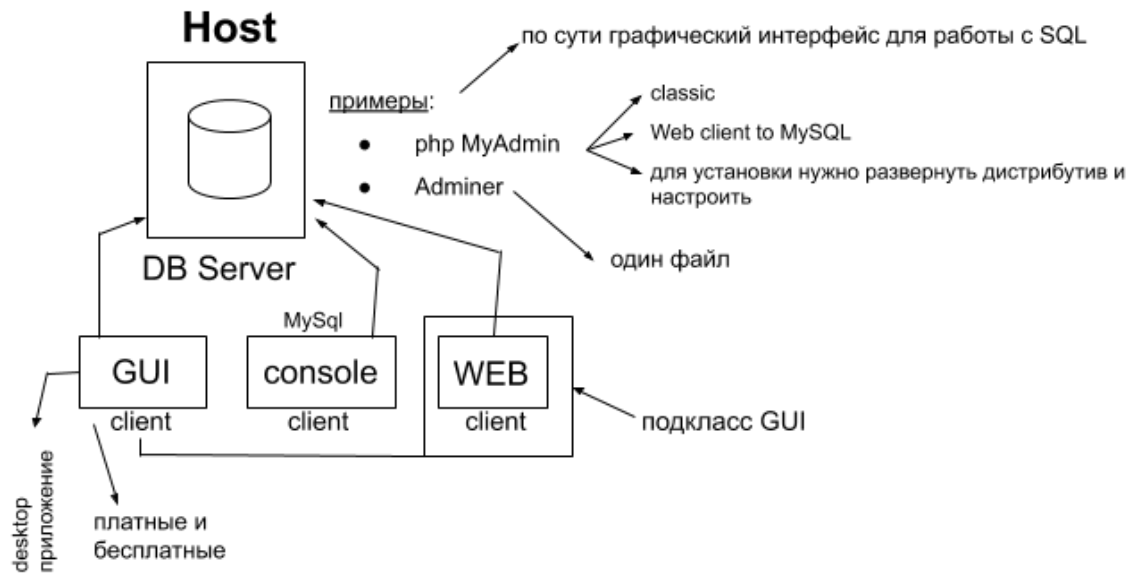
CSV Comma-Separated Values;

XML eXtensible Markup Language;

YML (YAML) Yet Another Markup Language;

Лекция 6. Введение в MySQL.

Structured Query Language SQL — формализованный язык для запросов к БД.

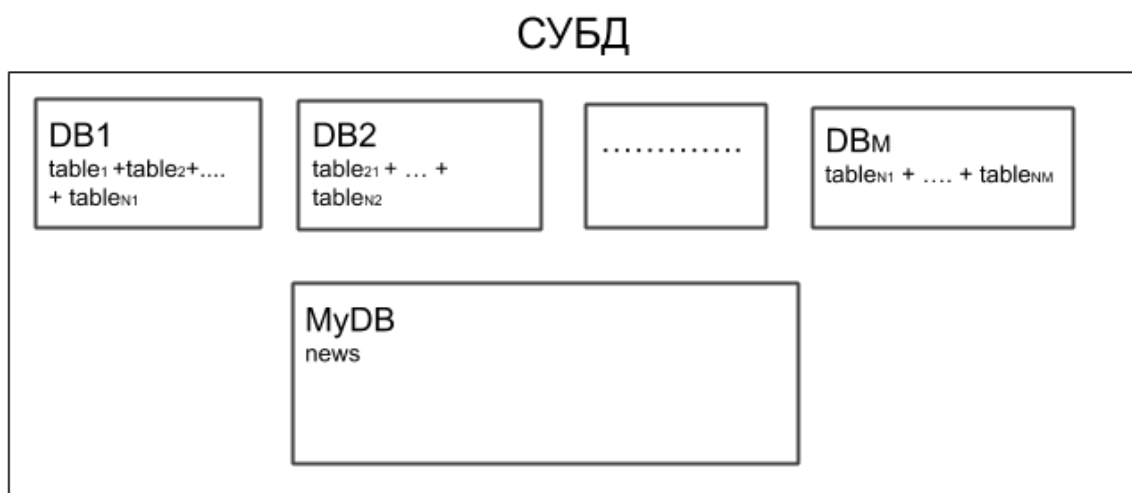


GUI graphical user interface;

1. Создание БД с именем MyDB:

mysql>create MyDB;

в конце строки обязательно ставим ;



2. Переключение на БД MyDB:

mysql>use MyDB;

3. Создание таблиц:

**mysql>CREATE TABLE news(news_id INT NOT NULL AUTO_INCREMENT,
heading VARCHAR(48), body TEXT, date DATE, author_n VARCHAR(48),
author_eml VARCHAR(48), PRIMARYKEY(news_id));**

INT целочисленный тип;

NOT NULL никогда не может быть пустым;

AUTO_INCREMENT автоматическое определение id;

Caps lock не имеет значения, в данном примере он использовался для удобства. Мы отделяем названия таблицы и ячеек от команды.

VARCHAR(48) тип данных, число в скобках означает, что в строке 48 символов;

хранится так

DATE	****_**_**
	год месяц день

После **,** в запросе лучше ставить пробел;

Таблица, которую мы создали теперь имеет такой вид:

news					
news_id	heading	body	date	author_n	author_eml

4. Ввод данных в таблицу:

**mysql>INSERT INTO news(NULL, 'Zagolovok', 'Text novosti', '14-03-2019',
'Ivanov', 'ivanov@ivanov.ru');** НЕ ВЕРНЫЙ ФОРМАТ ДАТЫ!

5. Редактирование записи:

mysql>UPDATE news SET date = '2019-03-14' WHERE news_id = '1';

выполнилась перезапись данных и таблица выглядит так:

news					
news_id	heading	body	date	author_n	author_eml
1	Zagolovok	Text novosti	2019-03-14	Ivanov	ivanov@ivanov.ru

При заполнении полей ставим одинарные кавычки для того, чтобы позже не запутаться в них, когда будем вставлять sql запрос, оформленный уже в двойные кавычки, в код php.

6. Удаление записи таблицы:

A) **DEETE FROM news where news_id = '4';**

B) **DELETE FROM news WHERE author_email LIKE 'ivanov@ivanov.ru';**

C) **DELETE FROM news WHERE author_email LIKE '%Ivanov%';**

Удаление записи с news_id = 4;

7. “Достать” данные из таблицы.

A) **SELECT * (все поле) FROM news;**

B) **SELECT news_id, heading FROM news;**

C) **SELCET * FROM news WHERE news_id = “1”;**

D) **SELECT * FROM news WHERE author_name LIKE 'Ivanov';**

E) **SELECT * FROM news WHERE author_name LIKE 'Ivanov %';**

F) **SELECT * FROM news WHERE author_name LIKE '%Ivanov%';**

Еще можно искать так: **'%vano%'**

LIKE сравнение по общей массе, можно просто ставить =

People		
int	char	text
id	name	text
1	Ivan	Tester
2	Nikolai	Programmer
3	Petr	CEO

CREATE TABLE People (id INT NOT NULL AUTO_INCREMENT, name

CHAR (20), text TEXT, PRIMARY KEY (id));

INSERT INTO People (NULL, 'Ivan', 'Tester');

INSERT INTO People (NULL, 'Nikolai', 'Programmer');

INSERT INTO People (NULL, 'Petr', 'CEO');

Теперь удалим строку 2, в строке 1 изменим text на Manager, а в строке 3 на Director.

UPDATE People SET text = 'Manager' WHERE id = '1';

UPDATE People SET text = 'Director' WHERE id = '3';

DELETE FROM People WHERE id = '2';

id	name	text
1	Ivan	Manager
3	Petr	Director

Связь таблиц.

Таблицы удобно связывать между собой, когда значения одного поля одинаково в нескольких строках и т.п. Пусть есть такая таблица:

Иванов	Иван	Иванович	бухгалтерия	Бухгалтер
Петров	Петр	Петрович	отдел сбыта	Менеджер
Сидоров	Николай	Степанович	отдел снабжения	Менеджер
Александров	Михаил	Альбертович	отдел качества	Бухгалтер
Николаев	Святослав	Владимирович	плановый отдел	Нанотехнолог
Николаев	Дмитрий	Николаевич	плановый отдел	Менеджер

В полях “отдел” и “должность” есть повторяющиеся “общие” значение (их можно шаблонно применять к любому сотруднику). В таком случае будет удобно вынести все возможные должности и отделы в отдельные таблицы, к которым мы будем обращаться через их id.

db.php

<?php

```
function get_val($table, $id, $cont)
```

```
{
```

```
    if($table == "otd") //отдел
```

```
    {
```

```
        $res = mysql_query("SELECT name FROM otd WHERE id_otd = ' ".$id." ', $cont);
```

```
        //оберем значение id из таблицы otd
```

```
    } else if($table == "dolj") //должность
```

```
    {
```

```
        $res = mysql_query("SELECT name FROM dolj WHERE id_dolj = ' ".$id." ',
```

```
$cont);
```

```
    }
```

```
    while($arr = mysql_fetch_array($res) )
```

```
    {
```

```
        return $arr['name'];
```

```
    }
```

```
}
```

```
function get_main_table($cont, $id)
```

```
{
```

```
    if($id==0)
```

```
    {
```

```
        $res = mysql_query("SELECT * FROM db", $cont);
```

```
    } else if($id>0)
```

```
    {
```

```
        $res = mysql_query("SELECT * FROM db WHERE id = ' ".$id." ', $cont);
```

```
    }
```

```
    echo "<TABLE border = '1'>";
```

```
    while($table = mysql_fetch_array($res)
```

```
    {
```

```
        echo
```

```
"<tr><td>".$table['f']. "</td><td>".$table['i']. "</td><td>".$table['o']. "</td><td>".get_val("o
```

```

td", $table['id_otd'], $cont)."</td><td>". get_val("dolj", $table['id_dolj'],
$cont)."</td></tr>";
}
echo "</TABLE>";
}
include("./lib.php");
$cont = mysql_connect("localhost", "root", "123");
if(!mysql_select_db("MyDB", $cont) )
{
    die( );
} else
{
    echo "String.... <br>";
    get_main_table("0", $cont); //вместо 0 может стоять $_GET['id'];
}
?>

```

Данный код выводит всю таблицу полностью. Чтобы можно было выводить значения соответствующих id, которые мы принимаем, нужно дописать эту часть кода:

```

else
{
    if(isset($_GET['id']) )
    {
        get_main_table($_GET['id'], $cont);
    } else get_main_table("0", $cont);
}

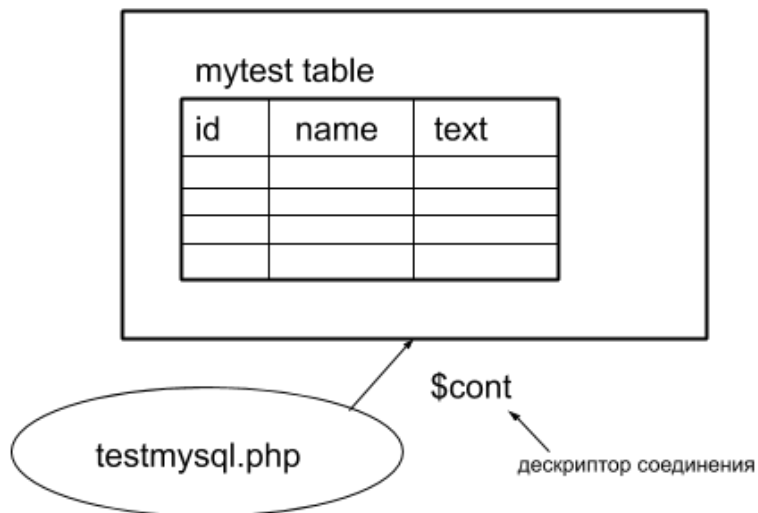
```

PHP + MySQL

У нас есть БД MyTest DB

Login: testuser

Password: testpass



testmysql.php

```
<?php
```

```
$cont = mysql_connect("localhost", "testuser", "testpass") or die("Can notconnect to DB");  
//дескриптор соединения с БД;  
mysql_connect_db("MyTest DB", $cont); //соединяемся с БД  
if($sql_result = mysql_query("SELECT * FROM mytest table", $cont) )  
{  
    echo "<table border = \"1\" >";  
    while($arr = mysql_fetch_array($sql_result) )  
    {  
        echo "<TR>";  
        echo "<TD>".$arr['id'].</TD>";  
        echo "<TD>".$arr['name'].</TD>";  
        echo "<TD>".$arr['text'].</TD>";  
        echo "</TR>";  
    }  
}
```

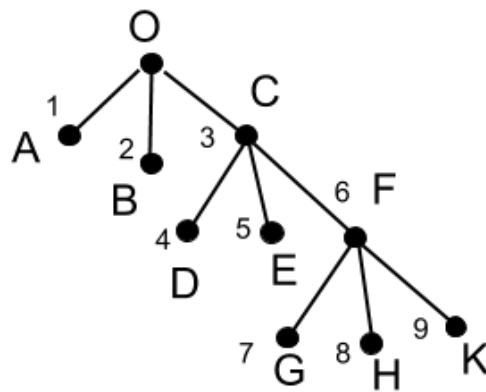
```

}
echo "</table>";
}
?>

```

Вместо 'localhost' может быть IP сервера при удаленном соединении.

Если не смешивать backend и frontend, то backend должен возвращать что-то в Json или т.п. и потом уже этот файл “оборачивать” в frontend.



| id | pid | name |
|----|-----|------|
| 1 | 0 | A |
| 2 | 0 | B |
| 3 | 0 | C |
| 4 | 3 | D |
| 5 | 3 | E |
| 6 | 3 | F |
| 7 | 6 | G |
| 8 | 6 | H |
| 9 | 6 | K |

Menu.php

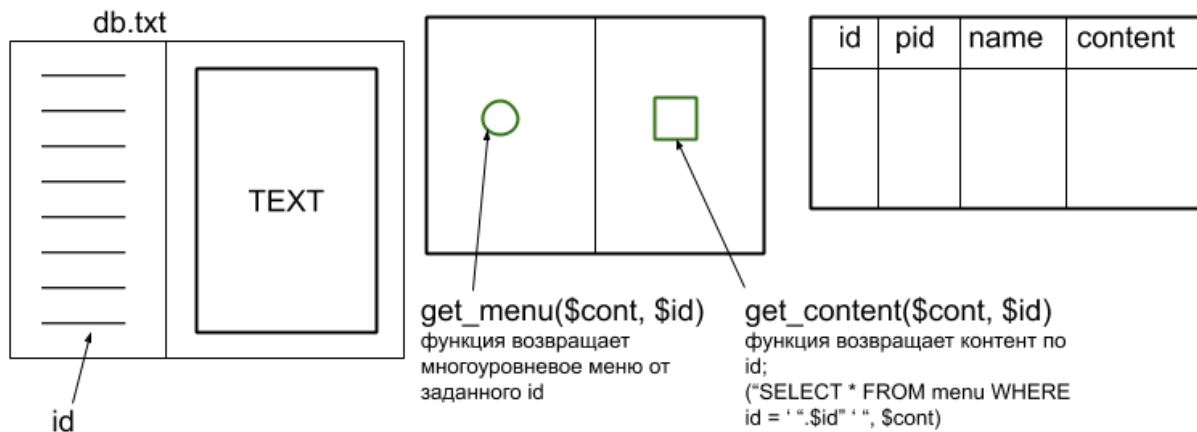
```

<?php
function get_children($id, $cont)
{
    $res = mysql_query("SELECT name FROM menu WHERE pid = ' ".$id." ', $cont);
    while($tree = mysql_fetch_array($res) )

```

```
{
    echo $tree['name'];
    get_children($tree['id'], $cont); //рекурсивный вызов функции
}
}
$cont = mysql_connect("lickalhost", "user", "pass");
mysql_select_db("MyDB", $cont);
get_children(0, $cont);
?>
```

Лекция 7.



Разметка: `<TABLE><TR><TD>○</TD><TD>□</TD></TR>`

db.php

```
<?php
include(".lib.php"); // "подключаем" содержимое файла
if( isset($_GET['id']) ) //проверяем не передано ли id
{
    $id = $_GET['id']; //если да – присваиваем переменной значение id
} else $id = 0; //если нет – присваиваем значение 0.
$cont = mysql_connect("localhost", "root", "123"); //создаем дескриптор
mysql_select_db("MyDB", $cont); //выбираем БД
//вывод таблицы с помощью тегов и функций
echo "<TABLE>";
echo "<TR><TD>".get_menu($cont, $id)."</TD><TD>".get_content($cont,
$id)."</TD></TR>";
echo "</TABLE>";
?>
```

lib.php


```

<?php
//функция возвращает строку, в которой лежит все, что нужно вывести в menu
function get_menu($cont)
{
    $sql = mysql_query($cont, "SELECT id, name FROM content");
    $str = " ";
    while($arr = mysql_fetch_array($sql))
    {
        $str = "<a href = \"db.php?id =\".$arr['id'].\">\".$array['name'].\"</a></br>";
    }
    return $str;
}
//функция возвращает массив значений, которые выводим в content
function get_content($cont, $id)
{
    $sql = mysql_query($cont, "SELECT diser FROM contenr WHERE id = ' ".$id." '");
    $arr = mysql_fetch_array($sql);
    return $arr['diser'];
}
?>

```

Теперь рассмотрим рекурсивный вывод таблицы

db2.php

```

<?php
include(".lib.php");
$cont = mysql_connect("localhost", "root", "123");
mysql_select_db("MyDB", $cont);
session_start(); //начинает новую сессию, либо возобновляет существующую
if( isset($_GET['id']) )
{
    $id = $_GET['id'];
} else $id = 0;

```

```
echo "<link rel = 'stylesheet' href = 'style.css' >"; //подключаем файл style.css, в котором
содержатся данные о стилях страницы
```

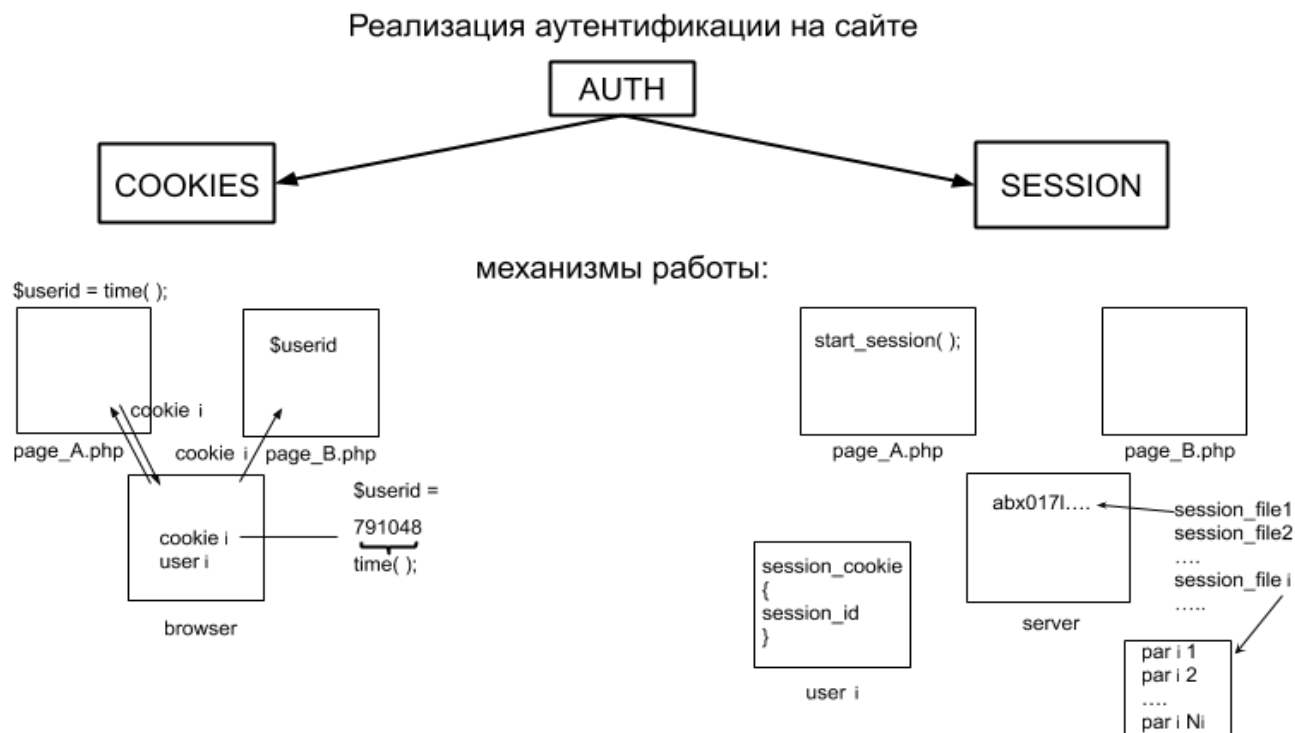
```
echo "<div class = menu>".get_menu($cont, 0, $id)."</div><div class = con-
tent>".get_content($cont, $id)."</div>";
?>
```

lib2.php

```
<?php
function get_menu($cont, $id, $id_menu)
{
    $sql = mysql_query($cont, "SELECT id, name FROM menu WHERE pid = '$id' ");
    $str = '<ul>';
    while($arr = mysql_fetch_array($sql))
    {
        if($id_menu == $arr['id'])
        {
            $str = "<li><strong><a href = \"db2.php?id
            =\".$arr['id'].\">\".$arr['name'].\"</a></strong></li></br>";
        } else
        {
            $str = "<li><a href = \"db2.php?id
            =\".$arr['id'].\">\".$arr['name'].\"</a></strong></li></br>";
        }
        $str = get_menu($cont, $arr['id'], $id_menu);
    }
    $str = "</ul>";
    return $str;
}
function get_content($cont, $id)
{
    $sql = mysql_query($cont, "SELECT content FROM menu WHERE id = ' ".$id." ' ");
    $str = " ";
    while($arr = mysql_fetch_array($sql) )
```

```
{  
  $str. = $arr['content'];  
}  
return $str;  
}  
?>
```

Лекция 8. Session and Cookie.



Session и cookies предназначены для хранения информации о пользователях при переходах между несколькими страницами. При использовании session данные сохраняются во временных файлах на сервере, а при использовании cookies – на устройстве пользователя и по запросу отсылаются браузером серверу.

Использование этих средств хранения очень удобно для интернет-магазинов, форумов и т.п., т.к. тут необходимо сохранять информацию о пользователях на протяжении нескольких страниц и своевременно предоставлять пользователю новую информацию.

Протокол HTTP является протоколом «без сохранения состояния», т.е. данный протокол не имеет встроенного способа сохранения состояния между двумя транзакциями (Когда пользователь открывает сначала одну страницу сайта, затем переходит на другую страницу этого же сайта, то основываясь только на средствах, предоставляемых протоколом HTTP невозможно установить, что оба запроса относятся к одному пользователю). Поэтому необходим метод, с помощью которого можно отслеживать информацию о пользователе в течение одного сеанса связи с web-сайтов. Одним из таких методов является управление сеансами при помощи предназначенных для этого функций. Важно, то сеанс представляет

собой группу переменных, которые, в отличие от обычных переменных, сохраняются и после завершения PHP-сценария.

Session. Пример работы.

http://students.yss.su/PSTGU/example/auth_sessions/

```
<form action="auth.php" method="POST"><br>
login <input type="text" name="login"><br>
<input type="submit">
```

http://students.yss.su/PSTGU/example/auth_sessions/auth.php

```
<?php
session_start();
if(isset($_POST['login']) )
{
$_SESSION['login']=$_POST['login'];
}
if(isset($_SESSION['login']) )
{
echo "session val:".$_SESSION['login'].<br>";
}
echo "session id:".session_id( );
echo "<a href=\"test.php\">test</a><br>";
echo "<a href=\"destroy.php\">destroy</a><br>";
?>
```

http://students.yss.su/PSTGU/example/auth_sessions/test.php

```
<?php
session_start();
echo $_SESSION['login'];
echo "<a href=\"auth.php\">back</a>";
```

```
?>
```

http://students.yss.su/PSTGU/example/auth_sessions/destroy.php

```
<?php
session_start( );
echo $_SESSION['login'];
unset($_SESSION['login']);
session_destroy( );
echo "<a href=\"auth.php\">back</a>";
?>
```

Cookie. Пример работы.

Cookies – это текстовые строки, хранящиеся на стороне клиента, и содержащие пары «имя-значение», с которыми связан URL, по которому браузер определяет нужно ли посылат cookies на сервер.

<http://students.yss.su/PSTGU/example/cookies/>

```
cookies example<br>
<form action="auth.php" method="POST"><br>
cookie val: <input type="text" name="val"><br>
<input type="submit" value="go">
```

<http://students.yss.su/PSTGU/example/cookies/auth.php>

```
<?php
if(isset($_POST['val']) )
{
    if(setcookie("val", $_POST['val'], time( ) + 60) )
```

```

{
    echo "cookie saved";
}else
{
    echo "cookie not saved";
}
}
echo "<a href=\"test.php\">test cookie</a><br>";
echo "<a href=\"destroy.php\">kill cookie</a><br>";
?>

```

<http://students.yss.su/PSTGU/example/cookies/test.php>

```

<?php
if(isset($_COOKIE) )
{
    echo "<pre>";
    print_r($_COOKIE);
    echo "</pre>";
}
if(isset($_COOKIE['val']) )
{
    echo "val = ".$_COOKIE['val'];
}
?>

```

<http://students.yss.su/PSTGU/example/cookies/destroy.php>

```

<?php
if(isset($_COOKIE['val']) )
{
    setcookie("val", $_COOKIE['val'], time( ) - 60*1000);
}

```

```

} else
{
    echo "val not present";
}
echo "<a href=\"auth.php\">back</a><br>";
echo "<a href=\"test.php\">test</a>";
?>

```

COOKIES		SESSION	
+	-	+	-
объем данных 20КБ	небезопасно	минимальный объем трафика	нет времени жизни
время жизни	данные хранятся на клиенте	более безопасно	
	трафик больший, чем при session	данные не хранятся на клиенте	

