LUCRUL CU SABLONUL MVC

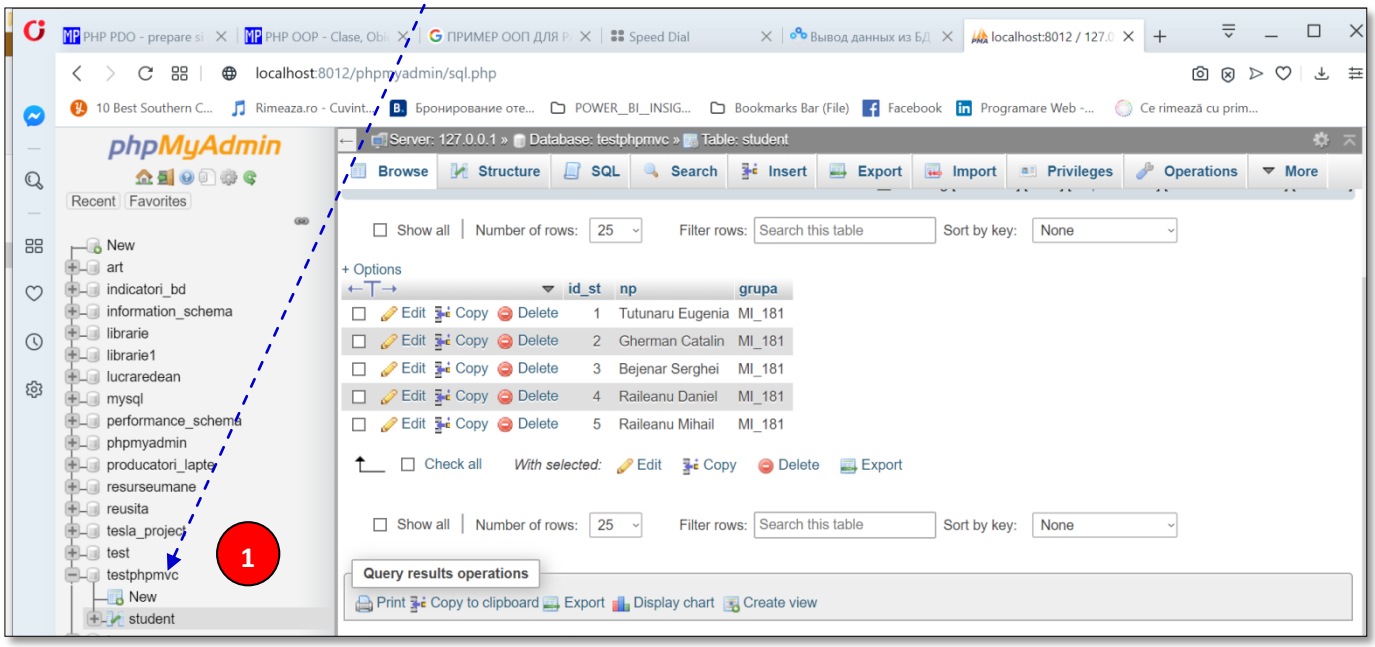
De la PHP POO la PHP POO+MVC

SCOPUL:

1. **Trecerea de la programarea procedural în PHP, cum a fost prezentată lucrarea de an, la dezvoltarea ei in paradigm/șablonul/patern-ul PHP POO+MVC.**
   Astfel, pentru lucrarea de an la MBD, Capitolul 1, urmeaza sa fie realizata lucrarea de an din semestrul precedent efectuata in PHP/HTML/CSS/MYSQL, mai numita si paradigma PROCEDURALĂ/"LINIARA", in paradigma PHP POO + MVC, utilizeind framework-ul Codeigniter.
2. **Prezentarea lucrării de an in PHP POO+MVC utilizeind framework-ul Codeigniter /vezi Partea a doua a Lab__2/.**

In exemplul ce urmeza este prezentat modul de trecere de la paradigma PROCEDURALĂ/"LINIARA", la paradigma PHP POO + MVC,

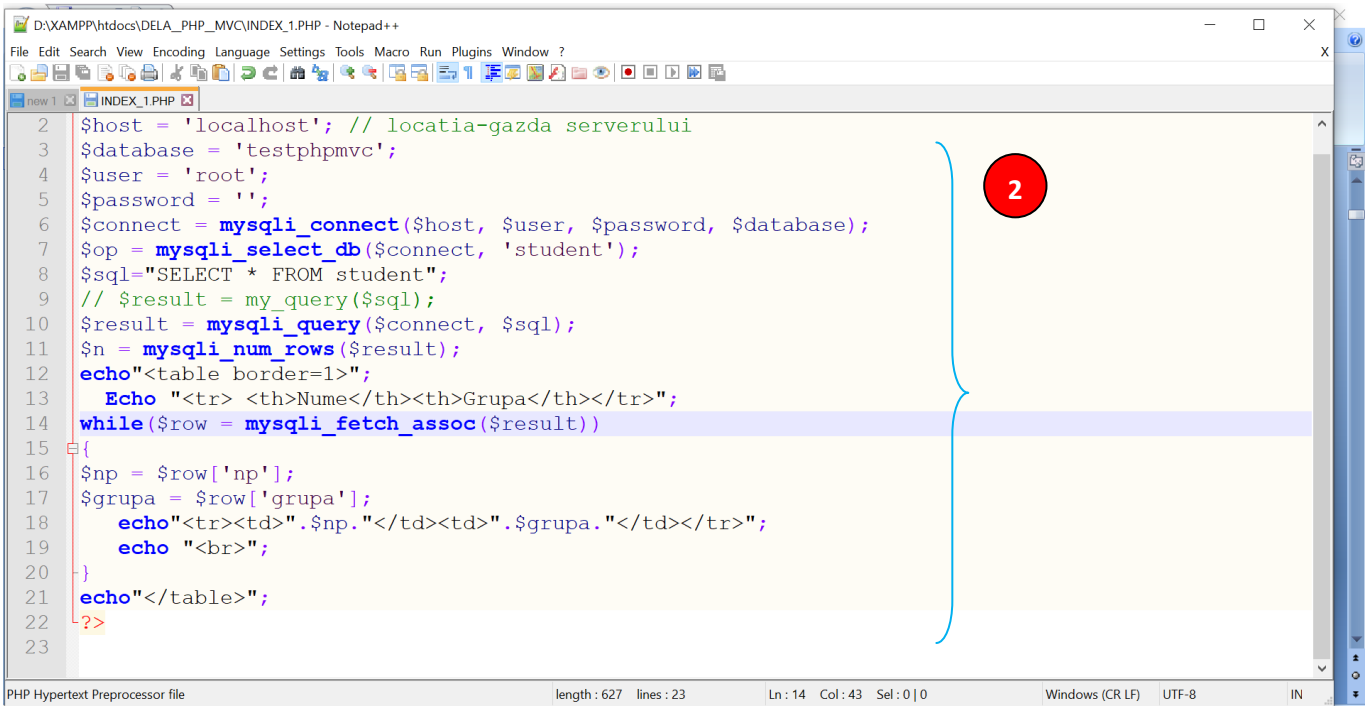
Baza de date, *testphpmvc*, se prezinta dupa cum urmeaza



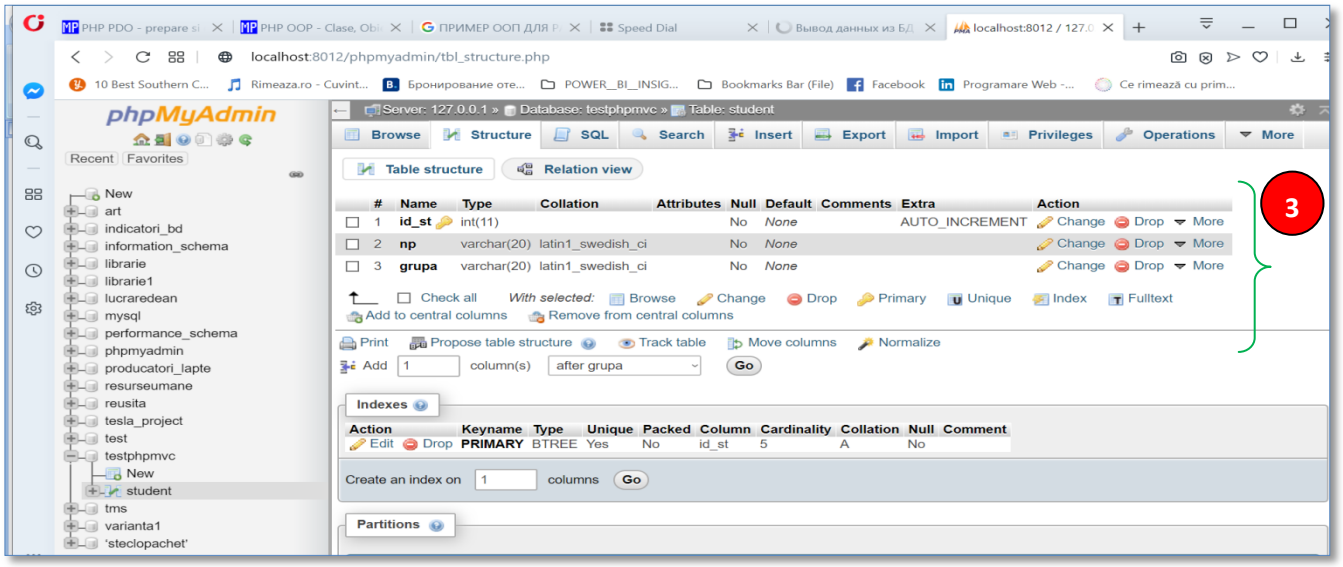
Codul **PHP/HTML/CSS/MYSQL** – **PROCEDURAL/"LINIAR",** de procesare a datelor, si anume, codul ce rezolvă **SARCINA**:

sa se prezinte lista studentilor grupei MI-201, din tabelul "STUDENT" a BD testphpmvc

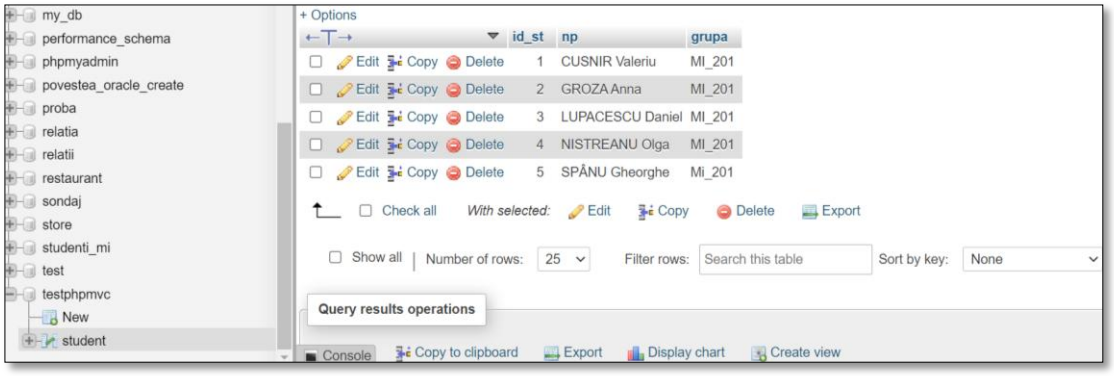
Acest rezultat este obtinut folosind scriptul **Index_1.php** după cum urmează:



Structura BD "**testphpmvc**" si tabelul ei unic "**student**" sunt prezentate aici,

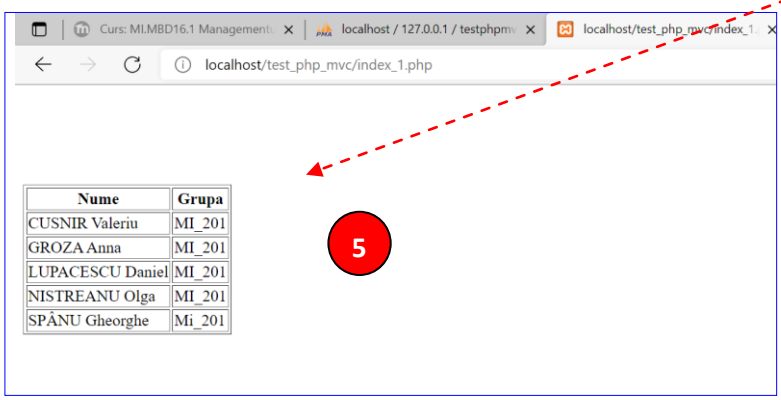


iar datele, sunt prezentate mai jos dupa cum urmeaza



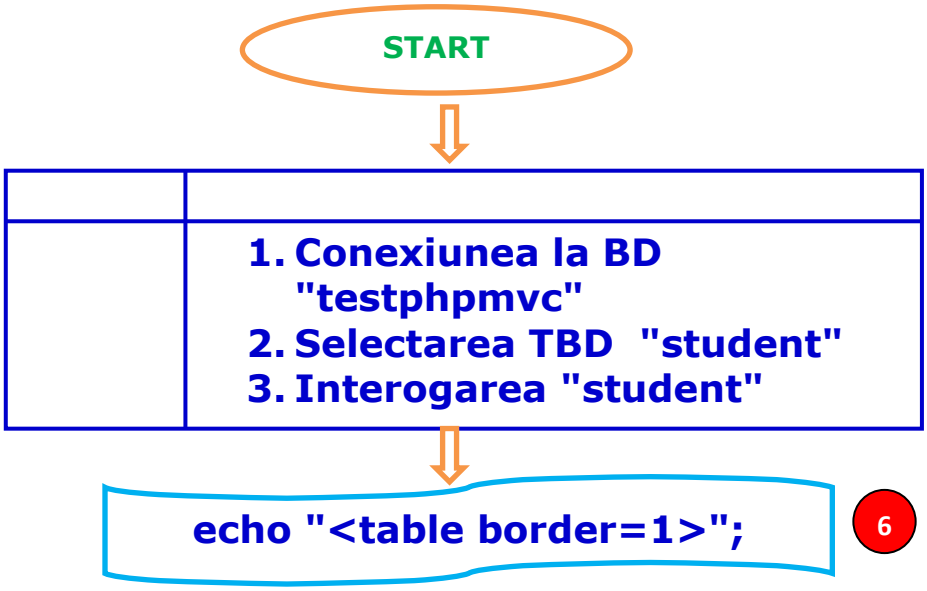
4

Rezultatul lucrului acestui script PROCEDURAL/"LINIAR", este

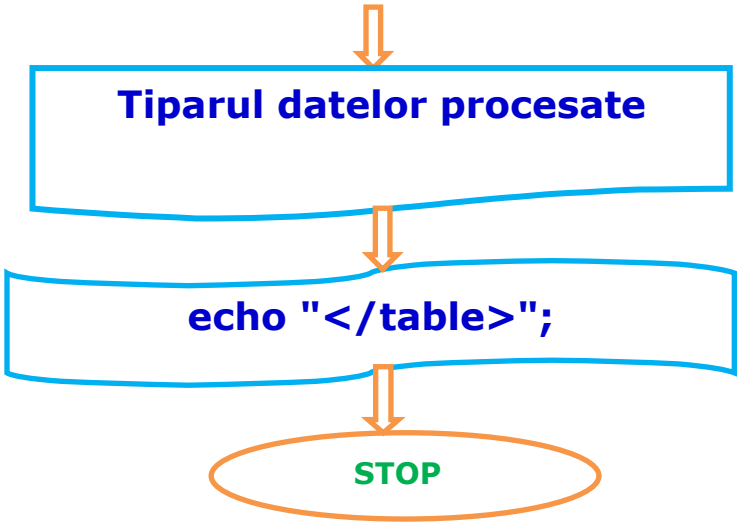


5

Schema bloc a aplicatiei PHP / PROCEDURAL/"LINIAR"/este



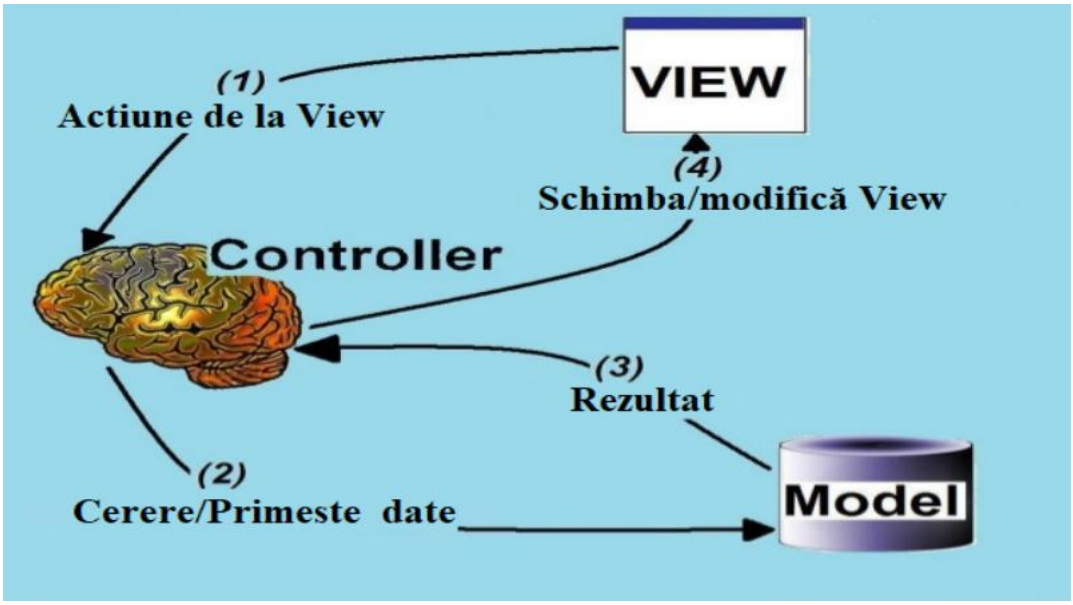
6

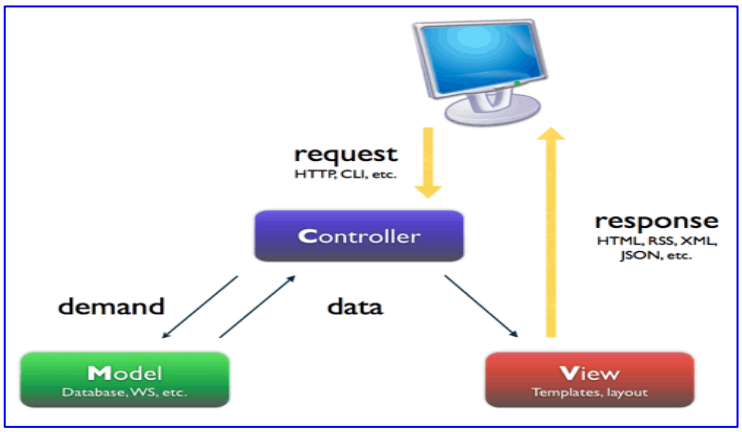


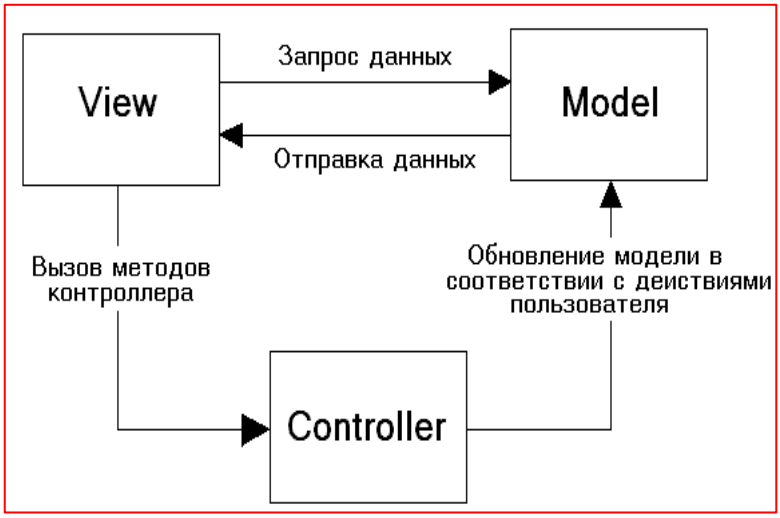
SARCINA

1) **Cum să prezentam codul scriptului initial prezentat in PHP – procedural/“liniar”, in paradigma/șablonul PHP POO+MVC, daca el este structurat în componente view.html, model.php și controller.php?**
2) **Cum să prezentam codul PHP POO in paternul MVC, folosind paradigma programarii orientate pe obiecte (POO) in PHP?**

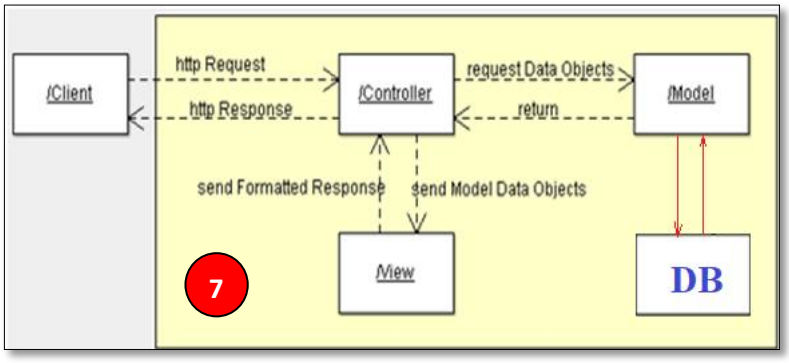
Reamintim paradigma MVC poate fi prezenta schematic după cum urmează



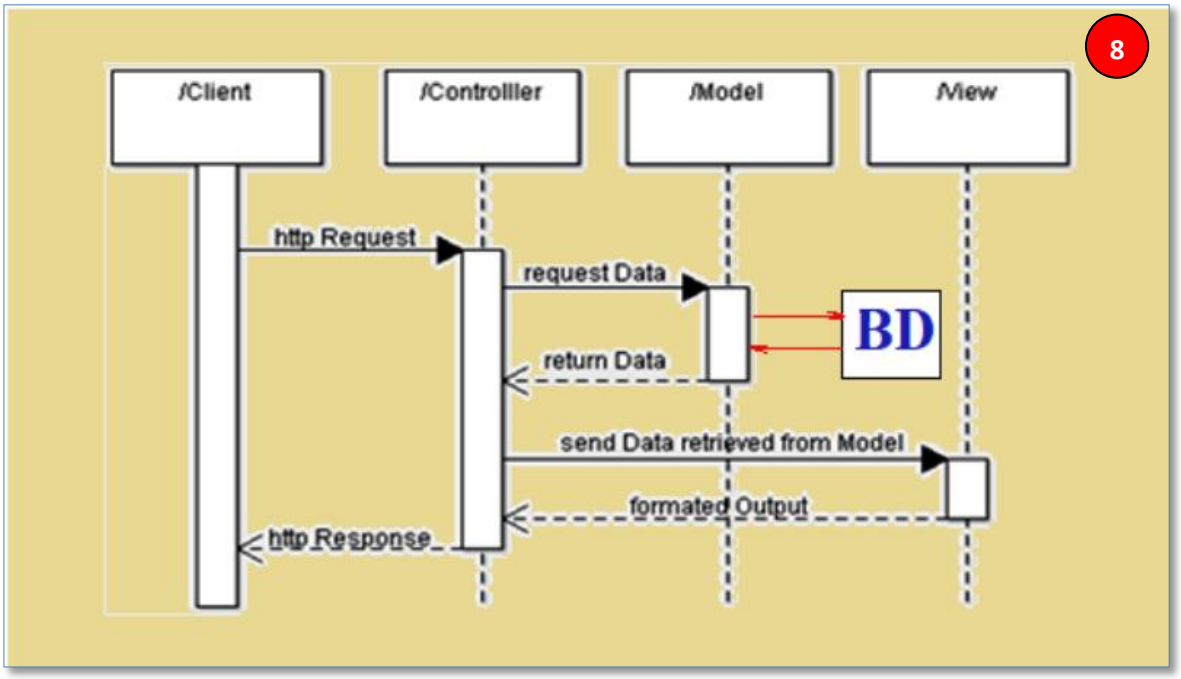

request
HTTP, CLI, etc.
Controller
response
HTML, RSS, XML, JSON, etc.
demand
data
Model
Database, WS, etc.
View
Templates, layout


View
Запрос данных
Model
Отправка данных
Вызов методов контроллера
Обновление модели в соответствии с деиствиями пользователя
Controller

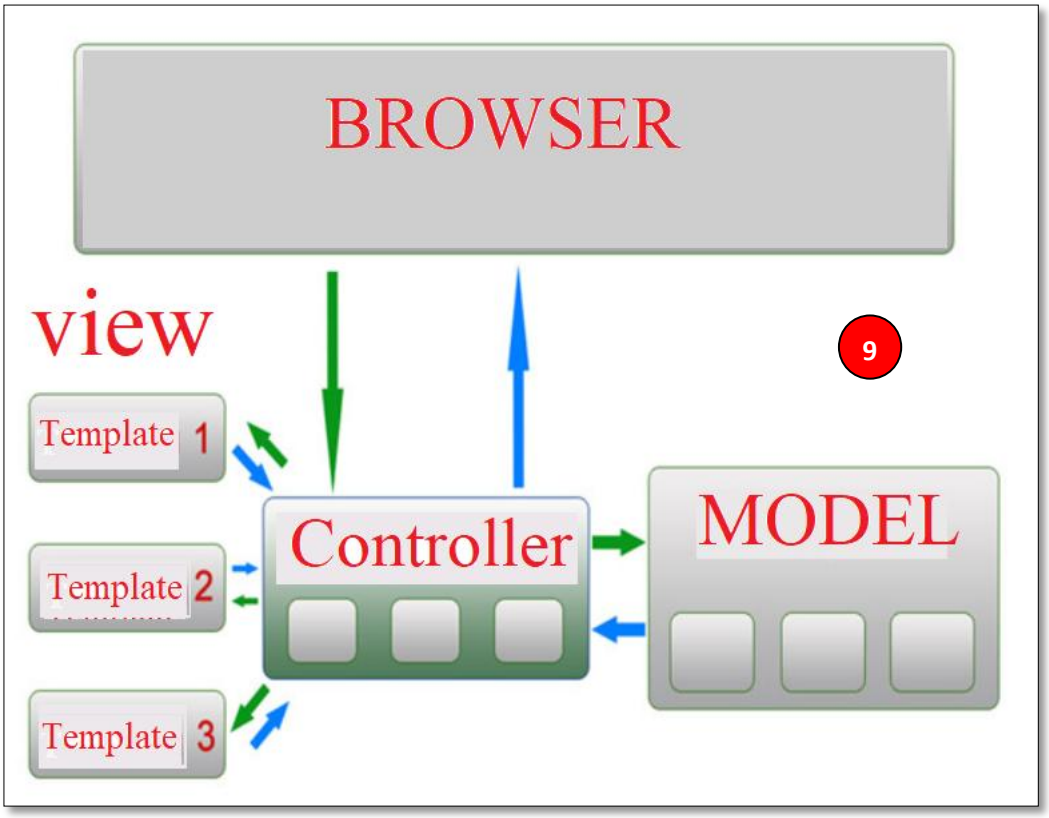

Пользователь
Вывод данных на экран
Отправка данных через точку входа
Шаблон
Контроллер
Выбор вида
Запрос на изменение данных
Запрос данных
Вид
Модель
Получение данных



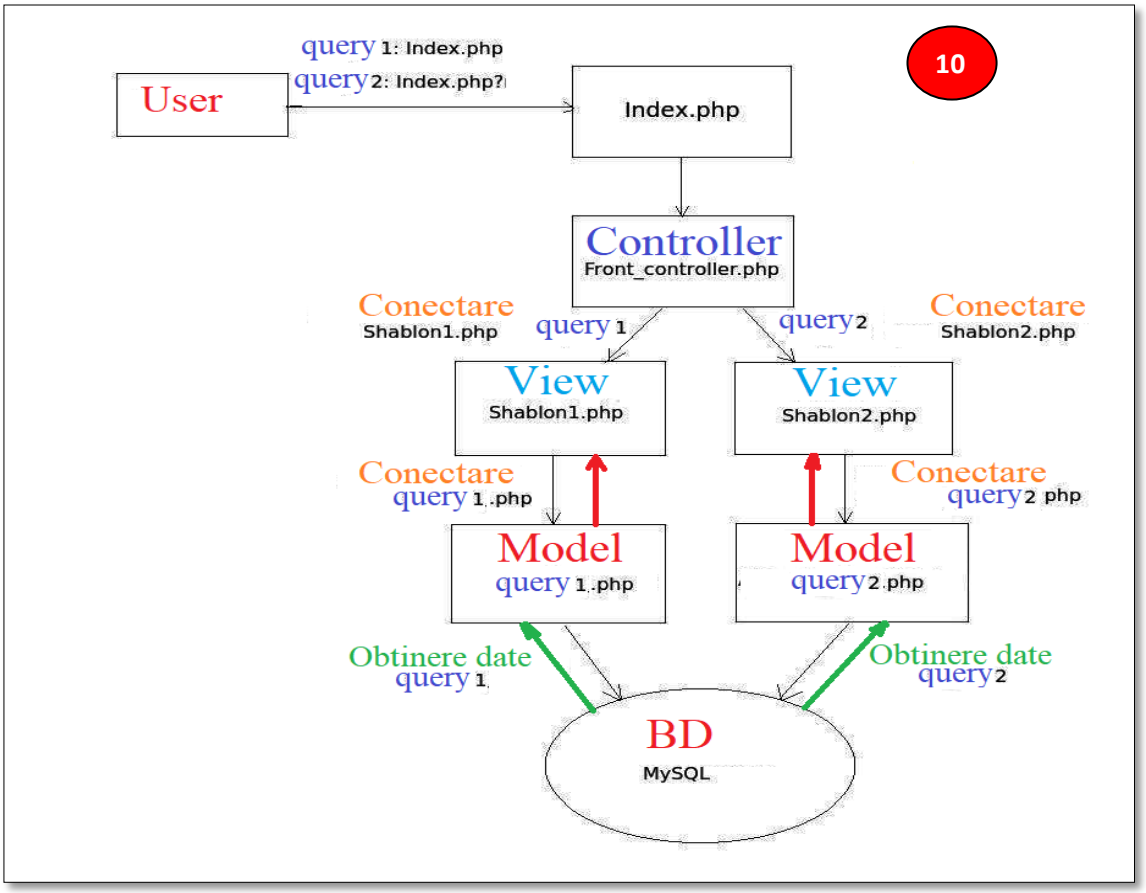
Sau prin următoarea diagramă de secvență **MVC**, in care putem **observa fluxul actiunilor** în timpul unei solicitări prin ***http*** de la browser:



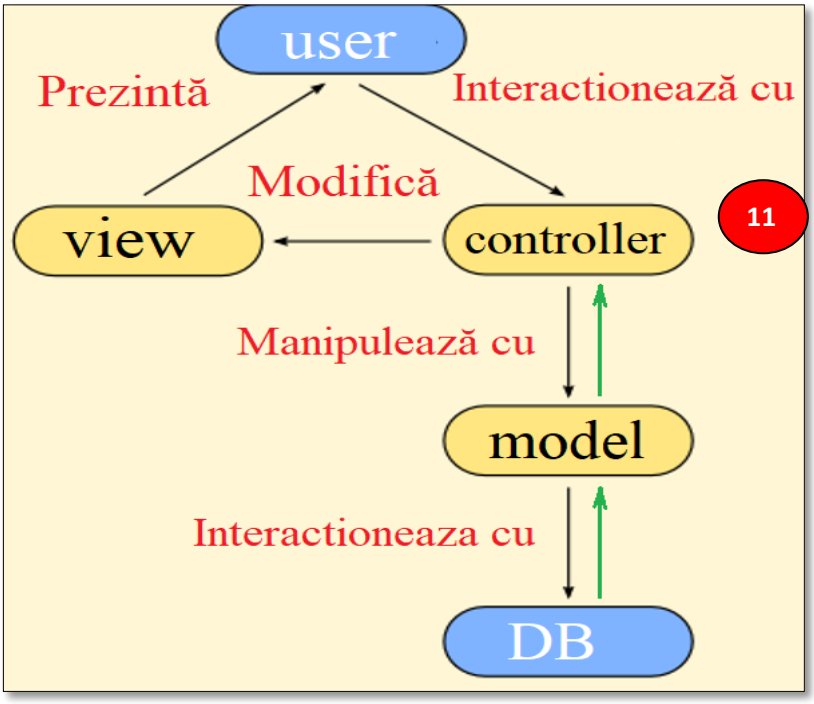
Care mai poate fi prezentata si după cum urmează



Ori in limbajul interogarilor, **atunci cind sunt 2 interogari**, putem schematic prezenta

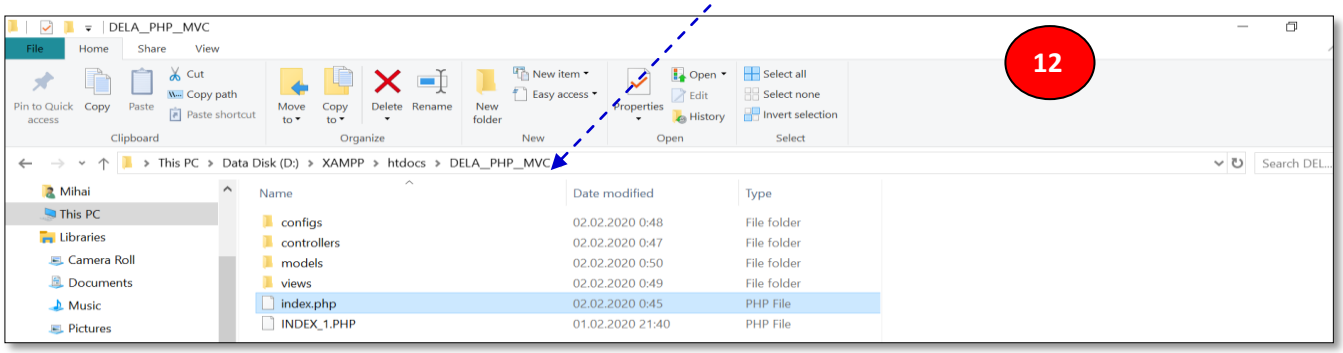


In continuare, pentru cazul nostru vom utiliza următoarea schema



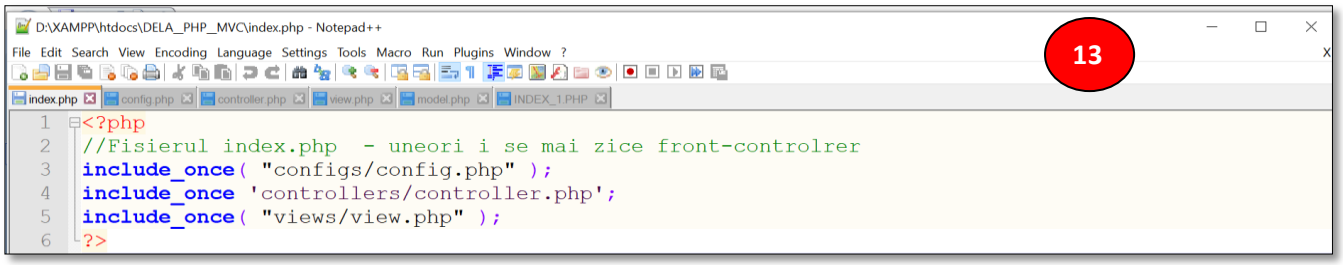
In cazul nostru iată solutia PHP POO + MVC

Structura mapei DELA__PHP__MVC /pentru Xampp/

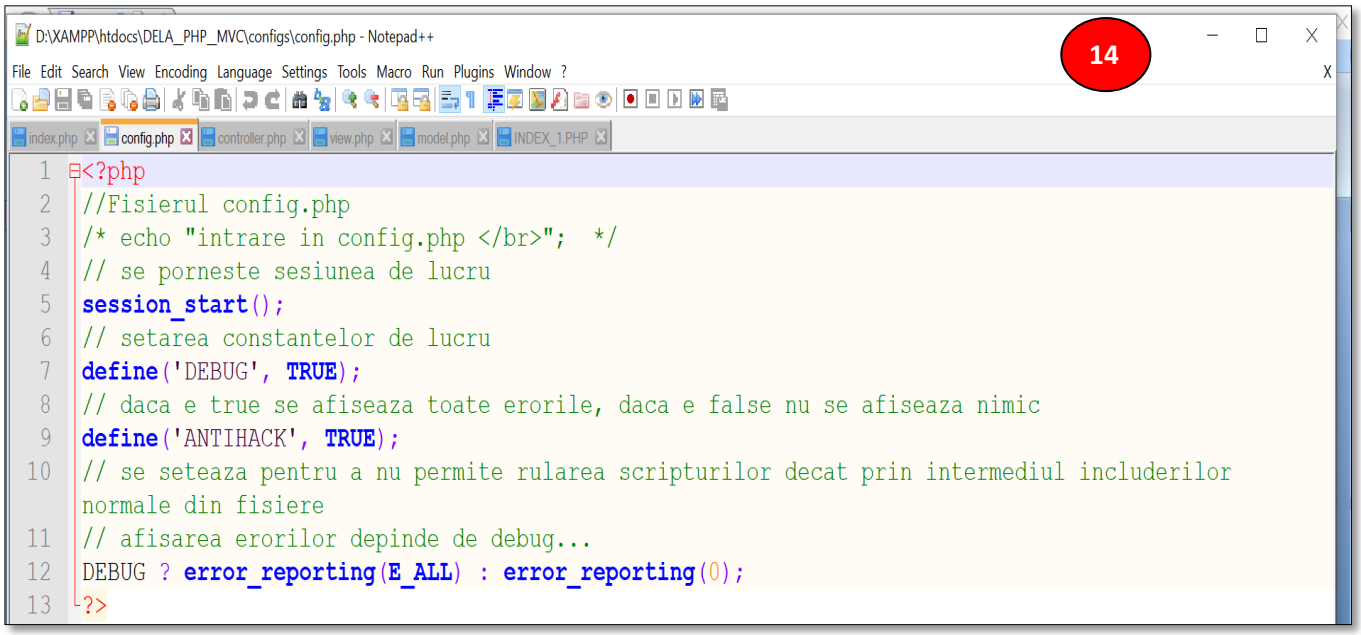


CU FISIERELE IN FIECARE DIN MAPE DUPA CUM URMEAZA:

Fisierul index.php - uneori i se mai zice front-controlrer



Fisierul config.php




```php
<?php
//Fisierul config.php
/* echo "intrare in config.php </br>";  */
// se porneste sesiunea de lucru
session_start();
// setarea constantelor de lucru
define('DEBUG', TRUE);
// daca e true se afiseaza toate erorile, daca e false nu se afiseaza nimic
define('ANTIHACK', TRUE);
// se seteaza pentru a nu permite rularea scripturilor decat prin intermediul includerilor normale din fisiere
// afisarea erorilor depinde de debug...
DEBUG ? error_reporting(E_ALL) : error_reporting(0);
?>
```

Fisierul controller.php


```php
<?php
//Fisierul controller.php
include_once ("models/model.php");
class GetDataFromDatabase extends database
{
    var $np = array();
    var $grupa = array();
    var $numeRows;
   function __construct()
   {
       parent::connect();
       self::get_data_from_db();
       parent::close();
   }
   function get_data_from_db() {
       $query = ("SELECT *FROM student ORDER BY id_st");
       $result = mysqli_query($this->connect,$query);
       $i = 0;
       while ($myrow = mysqli_fetch_array($result)){
           $this->np[$i]=$myrow['np'];
           $this->grupa[$i]=$myrow['grupa'];
           $i++;
       }
       $this->numeRows = $i-1;
   }
}
?>
```

Fisierul model.php

D:\XAMPP\htdocs\DELA_PHP_MVC\models\model.php - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

index.php | config.php | controller.php | view.php | model.php | INDEX_1.PHP

16

```php
<?php
//Fisierul model.php
class database
{
    var $connect;
    var $selectDatabase;

    public function connect()
    {
        $server = "localhost";
        $user = "root";
        $password = "";
        $db = "testphpmvc";
        $this->connect = mysqli_connect($server, $user, $password,$db);
        $this->selectDatabase = mysqli_select_db($this->connect,'student');
    }

    function close()
    {
        mysqli_close($this->connect);
    }
}
?>
```

Fisierul view.php

D:\XAMPP\htdocs\DELA_PHP_MVC\views\view.php - Notepad++

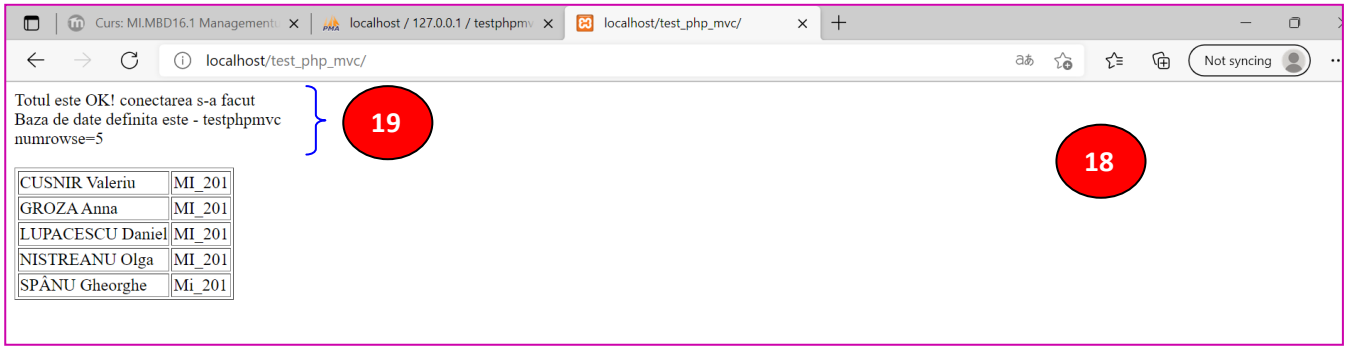
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

controller.php | model.php | view.php

17

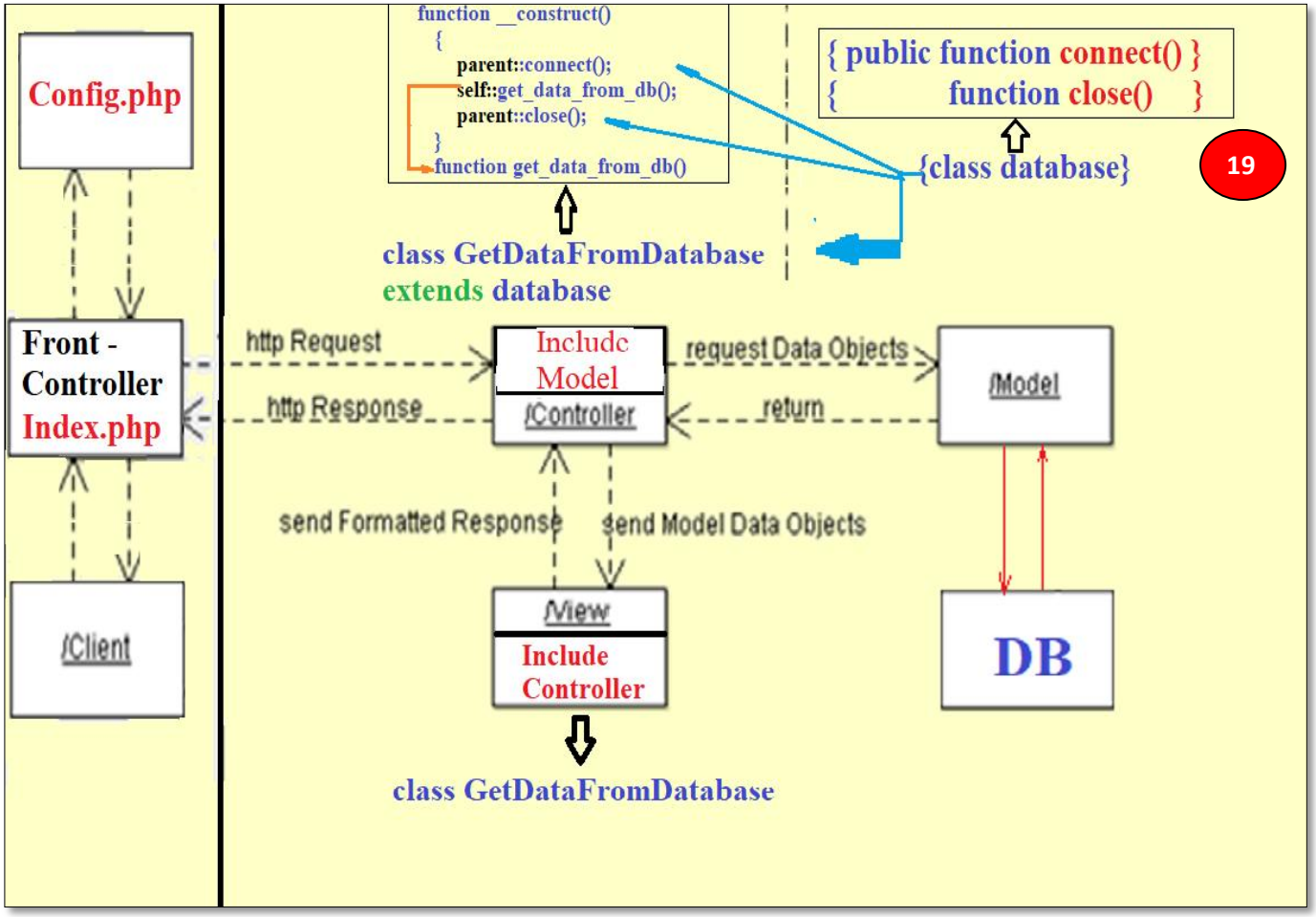
```php
<?php
// Fisierul view.php
error_reporting(E_ALL ^ E_DEPRECATED);
include_once 'controllers/controller.php';
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http:www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
  <title></title>
</head>
<body>
<table border="1">
    <?php
    $GetDataFromDatabase = new GetDataFromDatabase;
    $i = 0;
    echo"<table border=1>";
  Echo "<tr> <th>Nume</th><th>Grupa</th></tr>";
    while ($i <= $GetDataFromDatabase->numeRows) {
        echo "<tr><td>".$GetDataFromDatabase->np[$i]."</td><td>".$GetDataFromDatabase->grupa[$i]."</td></tr>";
        $i++;
    }
    echo"</table>";
    ?>
</table>
</body>
</html>
```

Rezultatul lansarii aplicatiei obtinute prin modelul MVC, cu ajutorul fisierului index.php, este...



NOTĂ: vă rog sa obțineți și informatia din 19. Va fi un punct adaugator!

Daca am utilize diagram de flux, atunci schematic pentru exemplul nostrum de mai sus modelul MVC poate fi prezentat dupa cum urmeaza:



Anexa

Material ajutator

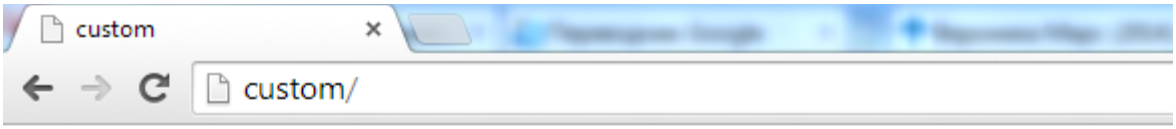
Пример использования циклов: отрисовка HTML таблицы на PHP

http://site-on.net/create/php/10-html-table-php

Здравствуйте уважаемые читатели блога **Site *on!*** В прошлой статье мы рассмотрели всё о **циклах** в PHP, но для закрепления полученных теоритических материалов, предлагаю выполнить практическую задачу, которая довольно часто встречается в повседневной жизни разработчика сайтов.

В наших с вами любимых CMS для отрисовки любой HTML таблицы используются циклы, по-другому никак. Например, у товара (смартфона) есть характеристики: ширина, высота, глубина, цвет, размер дисплея, объём памяти и тд. Все эти свойства обычно отрисовываются в виде HTML таблицы, это очень удобно и хорошо, ровно выглядит.

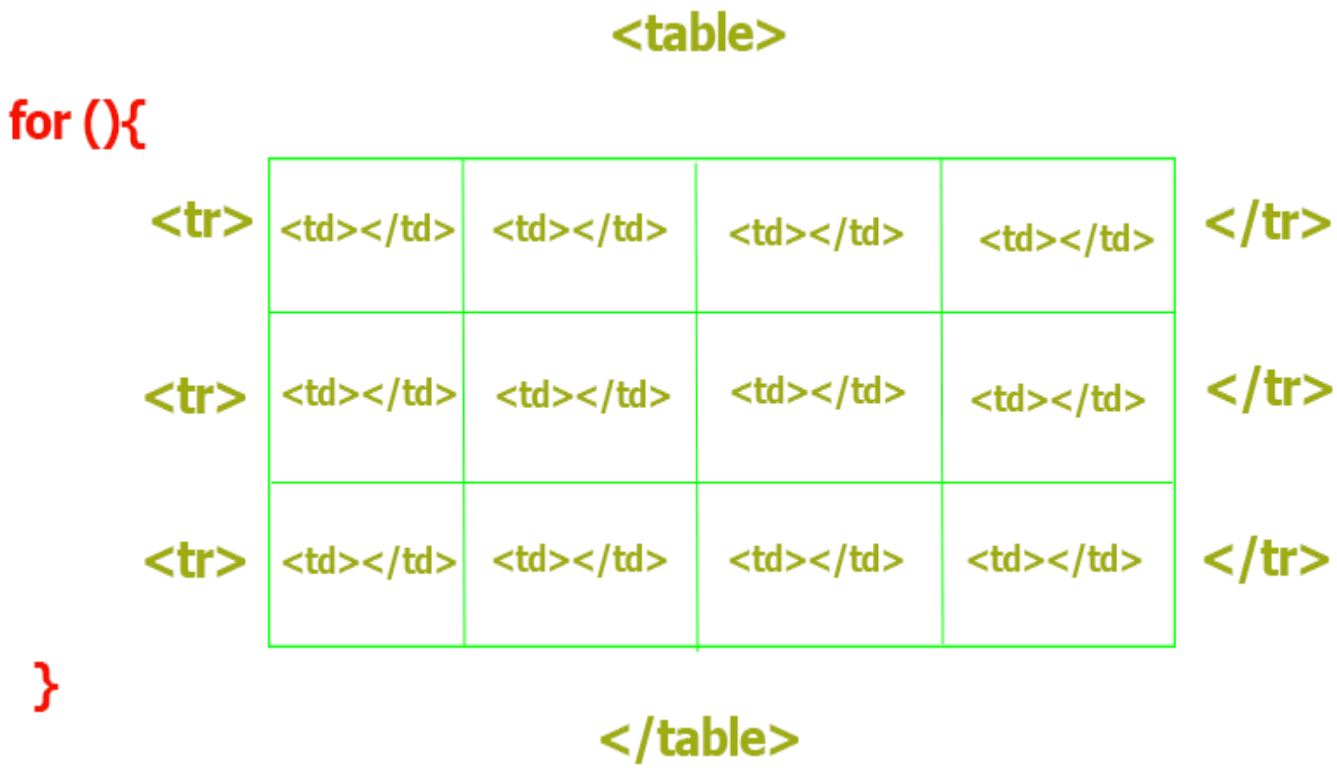
Чтобы понять, как это делается, сегодня мы решим такую простую задачу, как создание таблицы умножения:



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40
3	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48	51	54	57	60
4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68	72	76	80
5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100
6	12	18	24	30	36	42	48	54	60	66	72	78	84	90	96	102	108	114	120
7	14	21	28	35	42	49	56	63	70	77	84	91	98	105	112	119	126	133	140
8	16	24	32	40	48	56	64	72	80	88	96	104	112	120	128	136	144	152	160
9	18	27	36	45	54	63	72	81	90	99	108	117	126	135	144	153	162	171	180
10	20	30	40	50	60	70	80	90	100	110	120	130	140	150	160	170	180	190	200
11	22	33	44	55	66	77	88	99	110	121	132	143	154	165	176	187	198	209	220
12	24	36	48	60	72	84	96	108	120	132	144	156	168	180	192	204	216	228	240
13	26	39	52	65	78	91	104	117	130	143	156	169	182	195	208	221	234	247	260
14	28	42	56	70	84	98	112	126	140	154	168	182	196	210	224	238	252	266	280
15	30	45	60	75	90	105	120	135	150	165	180	195	210	225	240	255	270	285	300
16	32	48	64	80	96	112	128	144	160	176	192	208	224	240	256	272	288	304	320
17	34	51	68	85	102	119	136	153	170	187	204	221	238	255	272	289	306	323	340
18	36	54	72	90	108	126	144	162	180	198	216	234	252	270	288	306	324	342	360
19	38	57	76	95	114	133	152	171	190	209	228	247	266	285	304	323	342	361	380
20	40	60	80	100	120	140	160	180	200	220	240	260	280	300	320	340	360	380	400

Приступим

Для начала нужно придумать и понять алгоритм, которому мы будем следовать. Из чего состоит HTML таблица? Давайте я нарисую:



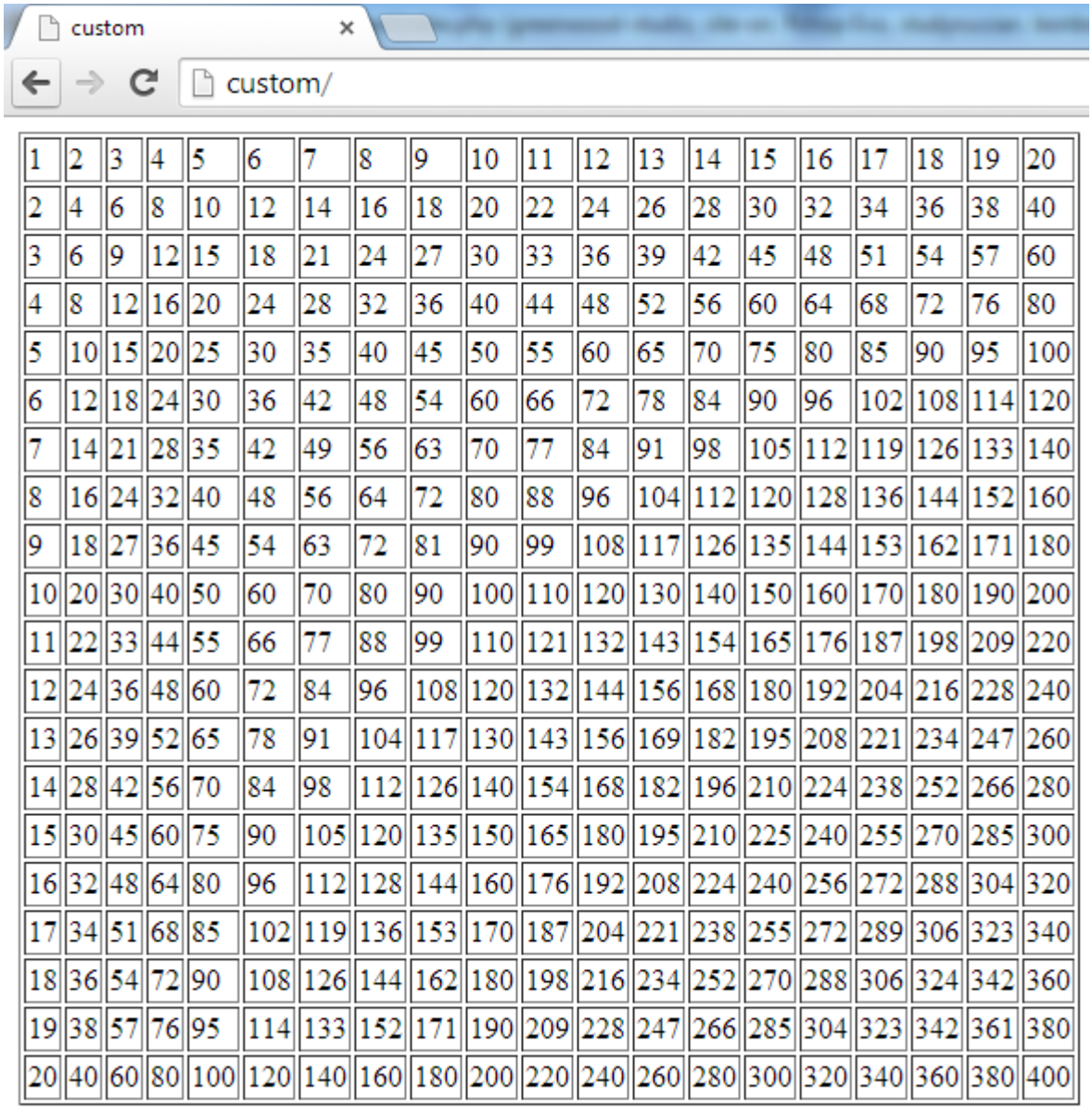
Как видим из рисунка, тег table встречается всего 1 раз, поэтому он явно должен быть вне цикла. Повторяется у нас только строка (tr) и внутри неё ячейки, они же столбцы (td). Всё, больше у нас ничего не повторяется.

Теперь нужно решить, какой цикл лучше использовать для данной задачи. Это классический вариант и лучшим решением будет цикл for, хотя можно использовать и while, если вы к нему привыкли больше. Но нам понадобится не один цикл, а два: первый будет рисовать и считать строки (tr), а второй столбцы (td).

```
<?php

$rows = 20; // количество строк, tr
$cols = 20; // количество столбцов, td

echo '<table border="1">';

for ($tr=1; $tr<=$rows; $tr++){ // в этом цикле счётчик $tr
   // следит за количеством строк и всегда равен текущему номеру строки.
   // То есть в начале $tr=1, так как в начале у нас 1 строка, затем
   // каждый раз прибавляем единицу, пока не дойдём до заданного количества
   // $rows.
   echo '<tr>';
   for ($td=1; $td<=$cols; $td++){ // в этом цикле счётчик $td аналогичен
                                   // счётчику $tr.
```

```
        echo '<td>'. $tr*$td .'</td>';
    }
    echo '</tr>';
}

echo '</table>';

?>
```

Готово! Результат:



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40
3	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48	51	54	57	60
4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68	72	76	80
5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100
6	12	18	24	30	36	42	48	54	60	66	72	78	84	90	96	102	108	114	120
7	14	21	28	35	42	49	56	63	70	77	84	91	98	105	112	119	126	133	140
8	16	24	32	40	48	56	64	72	80	88	96	104	112	120	128	136	144	152	160
9	18	27	36	45	54	63	72	81	90	99	108	117	126	135	144	153	162	171	180
10	20	30	40	50	60	70	80	90	100	110	120	130	140	150	160	170	180	190	200
11	22	33	44	55	66	77	88	99	110	121	132	143	154	165	176	187	198	209	220
12	24	36	48	60	72	84	96	108	120	132	144	156	168	180	192	204	216	228	240
13	26	39	52	65	78	91	104	117	130	143	156	169	182	195	208	221	234	247	260
14	28	42	56	70	84	98	112	126	140	154	168	182	196	210	224	238	252	266	280
15	30	45	60	75	90	105	120	135	150	165	180	195	210	225	240	255	270	285	300
16	32	48	64	80	96	112	128	144	160	176	192	208	224	240	256	272	288	304	320
17	34	51	68	85	102	119	136	153	170	187	204	221	238	255	272	289	306	323	340
18	36	54	72	90	108	126	144	162	180	198	216	234	252	270	288	306	324	342	360
19	38	57	76	95	114	133	152	171	190	209	228	247	266	285	304	323	342	361	380
20	40	60	80	100	120	140	160	180	200	220	240	260	280	300	320	340	360	380	400

В первом цикле мы отрисовываем tr, внутри него td. Этот код можно было бы назвать идеальным шаблоном отрисовки любой таблицы, но я знаю, как сделать ещё лучше. Незачем каждый раз напрягать PHP и делать вывод на экран (echo) после каждой итерации, гораздо лучше поместить всё в одно место (переменную) и сделать echo в самом конце, один единственный раз:

```
<?php

$rows = 20; // количество строк, tr
```

```
$cols = 20; // количество столбцов, td

$table = '<table border="1">';

for ($tr=1; $tr<=$rows; $tr++){
    $table .= '<tr>';
    for ($td=1; $td<=$cols; $td++){
        $table .= '<td>'. $tr*$td .'</td>';
    }
    $table .= '</tr>';
}

$table .= '</table>';
echo $table; // сделали эхо всего 1 раз

?>
```

Для этого мы использовали оператор **.=** присвоение через **конкатенацию**. Результат будет точно таким же. Но это ещё не всё. Нам нужно привести нашу таблицу к виду, как на самом первом рисунке в этой статье. То есть сделать первую строку и столбец полужирным и поставить зелёный фон. Такого результата можно добиться двумя способами:

- С помощью CSS3 (правильный способ);
- С помощью PHP (неправильный способ, но возьмём его, так как в этом разделе учим PHP);

Итак, делаем с помощью PHP:

```
<?php

$rows = 20; // количество строк, tr
$cols = 20; // количество столбцов, td

$table = '<table border="1">';

for ($tr=1; $tr<=$rows; $tr++){
    $table .= '<tr>';
    for ($td=1; $td<=$cols; $td++){
        if ($tr===1 or $td===1){
            $table .= '<th style="color:white;background-color:green;">'. $tr*$td .'</th>'; // вычислили первую строку или столбец
        }else{
            $table .= '<td>'. $tr*$td .'</td>'; // все ячейки, кроме ячеек из первого столбца и первой строки
        }
    }
    $table .= '</tr>';
}

$table .= '</table>';
echo $table; // сделали эхо всего 1 раз

?>
```

Красота :) Можете выводить таблицу хоть 100 на 100.