Управляващи структури: Оператори за разклонение, за организиране на цикъл, за предаване на управление и Други контролни структури

Виолета Божикова

Какво разглежда тази лекция?

- условни оператори (if, switch),
- работа с цикли (while, for, foreach)
- други контролни структури (include, require и др.)
- Много подкрепящи лекцията примери

Условни оператори (if, switch)

Структурата му може да се покаже както следва:

- **if (израз) блок_за_изпълнение**

Тук **израз** е всеки правилен PHP-израз. В процеса на обработка на скрипта, изразът се преобразува в логически тип. Ако в резултат на преобразуване стойността на израза е истина (True), то се изпълнява блок_за_изпълнение. В противен случай блок_за_изпълнение се игнорира.

Ако блок_за_изпълнение съдържа няколко команди, то той е длъжен да бъде заключен в фигурни скоби { }.

Правилата за преобразуване на израза към логически тип:

- За FALSE се считат:
  - Логическа стойност False
  - целочислена нула (0)
  - реална нула (0.0)
  - празен стринг и стринг "0"
  - масив без елементи
  - обект без променливи (подробно за обектите ще стане дума в следваща лекция)
  - тип NULL
- Всички останали значения се считат за TRUE. Разгледайте пример 1.

Оператор else, Оператор elseif

Съществуват няколко расширения на оператор *if*:

- Оператор ***else*** расширява *if* в случая когато проверявания *if* израз се явява неистинен:

if (израз) блок_за_изпълнение1

else блок_за_изпълнение2

Разгледайте пример 2.

- Еще един начин за разширение на *условния оператор if* е оператора ***elseif*** (*elseif* – това е комбинация на *else* и *if)*. Структурата на оператора *if*, разширен с помощта на оператори *else* и *elseif:*

if (израз1) блок_за_изпълнение1

elseif (израз2) блок_за_изпълнение2

…

else блок_за_изпълнениеN

Разгледайте пример 3.

Примери (1 и 2)

```
<?
$names = array("Ivan","Petar","Kamen");
if ($names[0]=="Ivan")
{   echo "Hello, Iva!";
    $num = 1;
    $account = 2000;
}
if ($num)
 echo "Ivan is the first in the list!";
$bax = 30;
if ($account > 100*$bax+3)
 echo "It's not shown on the screen, because
the condition is not true";
?>

//Hello, Iva!Ivan is the first in the list!
```

```
<?php
$a=1;
if ($a == 5):
    print "a is 5";
    print "...";
elseif ($a == 6):
    print "a is 6";
    print "!!!";
else:
    print "a is not 5, or 6";
endif;
?>
//a is not 5, or 6
```

Пример 5.

```
<?
$names = array("Ivan","Petar","Kamen");
if ($names[0]=="Ivan") {
    echo "Hello, Iva!";
    $num = 1;
    $account = 2000;
} else {
    echo "Hello, $names[0].
     We are looking for Ivan :(";
}
if ($num) echo "Ivan is the first in the
list!";
else echo "Ivan is not the first in the
list?!";
$bax = 30;
if ($account > 100*$bax+3)
    echo " It's not shown on the screen,
because the condition is not true ";
    else echo "You see this!";
?>

//Hello, Iva!Ivan is the first in the list!You
see this!
```

Пример 3,4

```
Пример 3:
<?
$names = array("Ivan","Petar","Kamen");
if ($names[0]=="Ivan") {
    // If the first name is Ivan
    echo "Hello, Iva!";
}elseif ($names[0] == "Petar"){
    // If the first name is not Ivan, but Petar
    echo "Hello, Peni!";
}elseif ($names[0] == "Kamen"){
    // If the first name is not
    // Ivan or Petar, but Kamen
    echo "Hello, Sena!";
}else {
    // If the first name is not Ivan,
    // or Petar or Kamen
    echo "Hello, $names[0]. And you are?";
}
?> //Hello, Iva!
```

```
<?php
$names = array("Ivan","Petar","Kamen");
if ($names[0]=="Ivan"):
?>
Hello, Iva!
<?php
endif;
?>
//Hello, Iva!
Пример 4. Използване на алтернативен синтаксис
```

Алтернативен синтаксис

- PHP предлага *алтернативен синтаксис* за някои свои управляващи структури, а именно за *if*, *while*, *for*, *foreach* и *switch*.
- Във всеки от случаите отварящата скоба трябва да се замени с двоеточие (:), а затварящата съответно с: endif;, endwhile; и т.н...
- Например, базовия синтаксис на оператора *if* може да се запише по такъв начин:

if(израз): блок_за_изпълнение endif;

- Смисълът е същия: Ако условието, записано в кръглите скоби на оператора *if* *е* истина, ще се изпълни целия код, от двоеточието «:» до команда endif;.
- Използването на такъв синтаксис е полезно при вграждане на php в html-код.

Оператор switch

- още една конструкция за разклонение - *switch*.
- В зависимост от това, какво **значение** има **„израз или променлива”**, *switch* превключва между различни **„блокове от оператори”**.
- Структурата на *switch* може да се запише по следния начин:

```
switch (израз или променлива){
case значение1:
    блок_от_оператори1
break;
case значение2:
    блок_от_оператори2
break;
...
default:

блок_от_оператори_по_подразбиране
}
```

Оператор switch

- *switch* много прилича на if...elseif...else или на набор от *if* оператори. Но, за разлика от *if*, тук значение на израза не се преобразува към логически тип, а просто се сравнява със значенията след *case* (значение1, значение2 и т.н.).
- Ако значение на израза съвпада с някакъв вариант, то се изпълнява съответстващия блок_от_оператори – от двоеточието до края на *switch* или до първото срещане на *break*, ако такъв се намери.
- Ако значение на израза не съвпада с нито един от вариантите, то изпълняваме блок_от_оператори_по_подразбиране (след default).
- Израз в *switch* се изчислява само един път, а в оператор *elseif* – всеки път, ето защо, ако изразът е достаточно сложен, то *switch* оператор работи по-бързо.
- Ако изпуснем оператор *break* програмата ще продължи изпълнението си до - *break.*
- За *switch*, както и за *if*, е възможен *алтернативен синтаксис*, при който отварящата в *switch* фигурна скоба се заменя с двоеточие, а затварящата с – endswitch; **Примери 6 и 6_2.**

Конструкции за цикъл в PHP

- В PHP съществуват няколко конструкции за цикъл: *while*, *do..while*, *foreach* и *for*.

while

- Структура:

while (израз) { блок_за_изпълнение } или

while (израз): блок_за_изпълнение endwhile;

- Командите в **блок_за_изпълнение** се изпълняват докато резултата от **израз** е **True** (и тук, както и в *if*, става привеждане на израза към логически тип).
- Значението на израза се проверява всеки път в началото на цикъла, така че дори и да настъпи промяна на стойността на израза в тялото на цикъла, цикълът няма да спре. **Пример 7.**

Конструкции за цикъл в PHP

do... while

- Цикъл ***do..while*** много прилича на *while*, с тази разлика, че истинността на израза се проверява в края на цикъла.
- Благодарение на това блок_за_изпълнение ще се изпълни поне веднъж.
- Структура:

do {блок_за_изпълнение} while (израз);

Пример 8.

Конструкции за цикъл в PHP

- ***For -*** Това е най-сложният оператор за цикъл в PHP. Много напомня същия оператор в C. Структура:

for (израз1; израз2; израз3) {блок_за_изпълнение} или

for (израз1; израз2; израз3): блок_за_изпълнение endfor;

  - **израз1** се изчислява безусловно един път – в началото на цикъла.
  - В началото на всяка итерация се изчислява **израз2**. Ако резултата от него е **True**, то цикълът продължава и се изпълнявав всички команди от блока_за_изпълнение. Ако **израз2** е False, то край на цикъла.
  - В края на всяка итерация (т.е. след изпълнение на всички команди от блока_за_изпълнение) се изчислява **израз3**.

- Всеки от изразите 1, 2, 3 може да е празен. Ако израз2 е празен, то това значи че цикълът ще се изпълнява неопределено време (в този случай PHP счита този израз за истинен винаги) - пример 10. Това не е така безполезно, както изглежда, тъй като цикълът може да бъде спрян с оператор break (пример 10). Например, всички четни цифри можем да изведем с for по следния начин (пример 9).

Пример 6.

```
<?
$names = array("Ivan","Petar","Kamen");
switch ($names[0]):
case "Ivan":
    echo "Hello, Iva!";
break;
case "Petar":
    echo "Hello, Peni!";
break;
case "Kamen":
    echo "Hello, Sena!";
break;
default:
    echo "Hello, $names[0].
    What is your name?";
endswitch;//Hello, Iva!
?>
```

Пример 7 (използване на while)

```
<?
//print all even numbers
    $i = 1;
    while ($i < 10) {
        if ($i % 2 == 0) print $i;
    // prin the numner, if even
        $i++;
    // increment $i with 1
    }
?> //2468
```

Пример 8 (използване на do…while)

```
<?
//the program prints 12.
$i = 12;
    do{
        if ($i % 2 == 0) print $i;
    // print the number if is even
        $i++;
    // increment the number
    }while ($i<10)
?> //12
```

Пример 9 (използване на for)

```
<?php
for ($i=0; $i<10; $i++){
    if ($i % 2 == 0) print $i;
    // print all even numbers: 02468
}
?>
```

Пример 10 (използване на break)

```
<?php
for ($i=0; ; $i++){
    if ($i>=10) break;
      // if $i>=10,
      // the cycle stops
    if ($i % 2 == 0) print $i;
      // If the number is even,
      // the program displays it
}
?>//02468
```

Примери

Пример 6_2.php код, вграден в html

```
<?
$names = array("Ivan","Petar","Kamen");
switch ($names[0]):
case "Ivan":
?>
<h1>Hello, Iva!<h1/>
<?
break;
case "Petar":
    echo "Hello, Peni!";
break;
case "Kamen":
    echo "Hello, Sena!";
break;
default:
    echo "Hello, $names[0].
     What is your name?";
endswitch;//Hello, Iva!
?>
```

Конструкции за цикъл в PHP

- Във for може да изпуснем всички изрази. В този случай просто не се задава начална стойност на брояча $i и той не се изменя в цикъла (Пример 11).
- В третия израз на конструкция *for* може да запишем няколко прости команди, разделени със запетая - виж Пример 12, където извеждаме всички цифри от 0 до 10.
- Ако блока съдържа само една команда или не съдържа команди (пак пример 12), то фигурни скоби може да няма!

Пример 11

```
<?php
$i=2; // initialization of the counter
for ( ; ; ){
    if ($i>=10) break;
       // if $i>=10,
       // the cycle stops
    if ($i % 2 == 0) print $i;
       // If the number is even,
       // the program displays it
$i++; // the counter increments
}
?> //2468
```

Пример 12

```
<?php
for ($i=0; $i<10; print $i, $i++)
?> //0123456789
```

Конструкции за цикъл в PHP

- И трите израза могат да включват повече от една инструкция – в този случай се разделят със запетая.
- Ако в израз2 има повече от една инструкция, се изпълняват всички, но се оценява само последната

```
<?
$total = 0;
for ($i = 0, $j = 1; $j++, $i <= 10; $i++, $j *= 2)
$total += $j;
echo "total=".$total; //total=6130
?>
```

foreach

- Еще една полезна конструкция (Примери 13). Тя е валидна от PHP4 нагоре и е предназначена изключително за работа с масиви.
- Синтаксис:

$array=array(“Ivan”,”Peter”,”Tom”);

foreach ($array as $value) {блок_за_изпълнение} или

foreach ($array as $key => $value) {блок_за_изпълнение}

- В първия случай се формира цикъл за обхождане на стойностите на всички елементи на масив, зададен чрез променлива **$array**.
- На всяка стъпка на цикъла, стойността на текущия елемент на масива се запазва в променлива **$value**, и вътрешния брояч на масива се придвижва с единица (така, че на следваща стъпка ще е видим следващ елемент на масива).
- Изпълнението на блока_за изпълнение ще се извърши толкова пъти, колкото са елементите на масива $array.
- При втората форма на запис, в допълнение - на всяка стъпка се запазва и ключа на текущия елемент в променлива **$key**.
- Когато ***foreach*** започва изпълнение, вътрешния указател на масива автоматичнО се установява на първия елемент.

foreach

- **!!!foreach() работи с копие на масива (Пример 14)**, така че ако в тялото на цикъла променяме елементите на масива, това няма да се отрази на оригиналния масив

```
Пример 14:
<?
$total = 0;
$names=array("Ivan","Tom","Iva" );
foreach ($names as $v)
{$v="Varna";
echo "<br>Hello $v";}
foreach ($names as $v)
{echo "<br>Hello $v";}
?>
        Резултат:
Hello Varna
Hello Varna
Hello Varna
Hello Ivan
Hello Tom
Hello Iva
```

Пример 13

```
<?php
$names = array("Ivan","Petar","Kamen");
foreach ($names as $val) {
    echo "Hello, $val <br>";
}
foreach ($names as $k => $val) {
    echo "Hello, $val !You are in the
list with number $k <br>";
}
?>
```

```
//Резултат (Пример 13 ):
Hello, Ivan
Hello, Petar
Hello, Kamen
Hello, Ivan ! You are in the list with number 0
Hello, Petar ! You are in the list with number 1
Hello, Kamen ! You are in the list with number 2
```

Пример 15. Оператор break

```
<?php
$i=1;
while ($i) {
    $n = rand(1,10);
    // генерираме произволно число
    // от 1 до 10
    echo "$i:$n ";
    // извеждаме номер на итерация и
    // генерираното число
    if ($n==5) break;
/* При число 5, се прекратява работата на
цикъла.*/
    echo "The cycle works<br>";
    $i++;
}
echo "<br>Number of iterations $i ";
?>
```

Резултата от работата на този скрипт е примерно:

```
1:6 The cycle works
2:7 The cycle works
3:10 The cycle works
4:5
Number of iterations 4
```

Примери

```
Пример 16. Оператор break
<?php
$i=1;
while ($i) {
$n = rand(1,10);
  // генерираме произволно число
  // от 1 до 10
  switch ($n){
    case 5:
      echo "<font color=blue>
        Изход от switch (n=$n)</font>";
break 1;
// прекратява се работата на switch
// (първо срещане на break в цикъла)
    case 10:
      echo "<font color=red>
       Exit from switch and
        while (n=$n)</font>";
    break 2;
// прекратява се работата на switch и
while
    // (два, съдържащи break цикли)
    default:
      echo "switch works (n=$n), ";
}
echo " while works - step $i <br>";
$i++;
}
echo "<br>Number of iterations $i ";
?>
```

```
switch works (n=8), while works – step 1
switch works (n=9), while works – step 2
switch works (n=6), while works – step 3
switch works (n=2), while works – step 4
Изход от switch (n=5) while works – step 5
Exit from switch and while (n=10)
Number of iterations 6
```

```
Пример 17. Оператор continue
<?php
$i=1;
while ($i<=4) {
    $n = rand(1,10);
    // генерираме произволно число
    // от 1 до 10
    echo "$i:$n ";
    // извеждаме номер на итерация и
    // генерираното число
if ($n==5) {
        echo "New iteration <br>";
        continue;
/* Ако се генерира число 5,
то започва нова итерация,
$i не се увеличава */
    }
    echo "The cycle works<br>";
    $i++;
}
--$i;
echo "<br>Number of iterations $i ";
?>
```

```
Резултат от работата на скрипт 17 е:
1:4 The cycle works
2:6 The cycle works
3:5 New iteration
3:2 The cycle works
4:6 The cycle works
Number of iterations 4
```

Оператори за предаване на управление - break

- Понякога се изисква незабавно да се излезе от цикъл. За това се използва оператор *break*.
- ***Break***
- Оператор *break* завършва изпълнението на текущия цикъл, било *for*, *foreach*, *while*, *do..while* или *switch (пример 15)*.
- *Break* може да се използва с числов аргумент
- Ако след оператора *break* се укаже число, то се прекъсва изпълнението на именно такова количество вложени цикли (пример16):
  - при $n=5 - прекратява се работата на switch (break 1)
  - при $n=10 - прекратява се работата на switch и while (break 2)

Оператори за предаване на управление - *continue*

- Понякога е нужно не напълно да се прекрати работата на цикъла, а само да се започне негова нова итерация. Оператор *continue* позволява да се пропуснат останалите инструкции от блока_за_изпълнение.
- *continue* може да се използва с числов аргумент, указващ колко съдържащи го управляващи конструкции трябва да завършат работа.
- Да заменим в пример 15 оператор break с continue (пример 17).
- Още – да ограничим броя на стъпки на цикъла на 3. Както се вижда от примера по време ма 3-та итерация се генерира числото 5, следва игнорирането на итерацията, тоест повторение на тази итерация.
- В PHP има една особеност на оператор *continue:* в конструкция *switch* той работи така както и *break*. Ако *switch* се намира вътре в цикъла и е нужно да започнем нова итерация на цикъла, следва да използваме continue 2.

Други контролни структури

exit - спира изпълнението на текущия скрипт;

exit може да приеме като параметър стойност:

- ако стойността е от тип низ, тя ще се отпечата, преди изпълнението на скрипта да спре.

```
<?php
$filename = '/user/peter/example.dat';
$file = fopen($filename, 'r')
   or exit("unable to open file ($filename)");
?>
```

Други контролни структури

- ако exit няма параметър или стойността на параметъра е 0, то имаме exit статус: **нормално изпълнение**
- ако exit има параметър от 1 – 254: статуси за грешка

```
<?php
//нормален изход от програмата
exit;
exit(); //die() - псевдоним на exit()
exit(0);
// изход с код за грешка
exit(1);
exit(0376); // код за грешка в осмичен код
?>
```

Други контролни структури

- **return()**

- Ако **return** е извикан от глобалния обхват, изпълнението на текущия скрипт се прекратява
- Ако **return** е извикан от включен (с require или include) скрипт, тогава контролът се връща на извикващия скрипт
  - Ако **return** е извикан от скрипт, включен с include, тогава стойността подадена на return() се връща като стойност на променливата, на която е присвоено извикването на include(), например:

```
/* included.php */
return(3) ...

/* including file */
$result = include 'included.php'; // $result == 3
```

- Ако е извикан от функция, **return**() прекратява изпълнението й и връща аргумента си като стойност от извикването на функцията.

Други контролни структури

- **include(), require()**

Пример:

```
<?php
include 'data.inc';
require 'b.php';
// ...
?>
```

- **Включват и изпълняват даден файл: т.е. вземат текста от специфицирания (от инструкция include/require) файл и го копират във файла, който използва include/require инструцията**

разлика между require и include

- Основната разлика между require и include е по отношение на реакцията им при възникване на грешка (липса на файл, грешка в самия файл).
- **При грешка, include извежда предупреждение, и работата на скрипта продължава.**
- **Грешка при require предизвиква фатална грешка на работата на скрипта и се прекратява изпълнението му**.

require/include - особености

- Кодът във включения файл наследява обхвата на действие на променливите на реда, в който става включването
- Ако включването е станало във функция, то все едно, че кодът от включения файл е бил част от дефиницията на функцията – променливите от него ще наследят локалния обхват на променливите на функцията (пример 20) .
- Кодът вътре във включения файл също трябва да бъде е php тагове
- require и include са специални езикови конструкции, ето защо при използване им в условни блокове трябва да се поставят във фигурни скоби.

Пример 21. Пример за include

```
<?php
/*  Това е грешен запис.Очаква се грешка.
*/
if ($condition) include("first.php");
else include("second.php");
// Сега е правилно.
if ($condition){ include("first.php"); }
else { include("second.php"); }
?>
```

Пример: include(), require()

- Нека във файл params.inc (пример 18) се съхранява набор от параметри и функции. Всеки път, когато са ни нужни тези параметри (функции), ние слагаме команда:

include 'params.inc'.

Включващият скрипт (пример 19) е **include.php**.

- Забележка: Резултатът ще бъде същия, ако заменим **include() с require().**

include(), require() - примери

Пример 19. include.php

```
<?php
include ("params.inc");
/* променливи $user и $today са зададени
в params.inc.*/

echo "Hello, $user!<br>";
    // извежда "Hello, Alla!"
echo "Today is $today";
?>
```

Резултат от работата на скрипт 19 е:

```
Hello, Alla!
Today is 25.11.11
```

Пример 18. params.inc

params.inc

```
<?php
$user = "Alla";
$today = date("d.m.y");
/* функция date() връща дата
и време (тук – датата е във формат
ден.месец.год) */
?>
```

Пример 20. include.php

Нека файл params.inc остане същия, а include.php се промени на:

```
<?php
function Footer(){
// функция Footer
    include ("params.inc");
/* включваме файл params.inc.
Сега параметрите могат да се използват
само във функцията */
    $str = "Today: $today <br>";
    $str .= "<a
    href='http://www.abv.bg/'>THE PAGE IS
CREATED BY $user</a>";
    echo "$str";
}
Footer();
// CALL Footer(). THE RESULT IS:
//Today: 08.07.11
// THE PAGE IS CREATED BY ...
echo "$user, $today";
/* извежда запетая, защото тези
променливи са видими само във функцията*/
?>
```

Резултат от работата на този скрипт е:

```
Today: 25.11.11
THE PAGE IS CREATED BY Alla
Notice: Undefined variable: user in
C:\xampp\htdocs\PhpProject1\index.php on line 24
Notice: Undefined variable: today in
C:\xampp\htdocs\PhpProject1\index.php on line 24
,
```

include()/require()

- **При включване с include, PHP обработва скрипта, анализирайки го ред по ред, докато не достигне до конструкция include. Стигайки до include, PHP прекратява транслирането на скрипта и превключва към указания в include файл. Видно е, че бързодействието се намалява, особено при по-голям брой включени с помощта на include файлове.**
- **Файловете с помощта на require се включват преди изпълнението на скрипта, тоест в момента на транслацията файлът вече е включен в скрипта. Ето защо, тя е по-бърза, в сравнение с include. За сметка на това обаче не може да се използва за динамично включване на файлове в кода на PHP скрипта. Затова е целесъобразно да се предпочете конструкция require там, където не е необходимо динамично включване на файлове в скрипта, а конструкция include да се използва само с цел динамично включване на файлове в кода на PHP скрипта.**

Други контролни структури

- include_once(), require_once()
- Това са оператори за еднократно включване на файл.
- Поведението им е идентично с това на require() и include(), с тази разлика, че ако файлът вече е бил включен, той няма да бъде включен повторно.

Други контролни структури

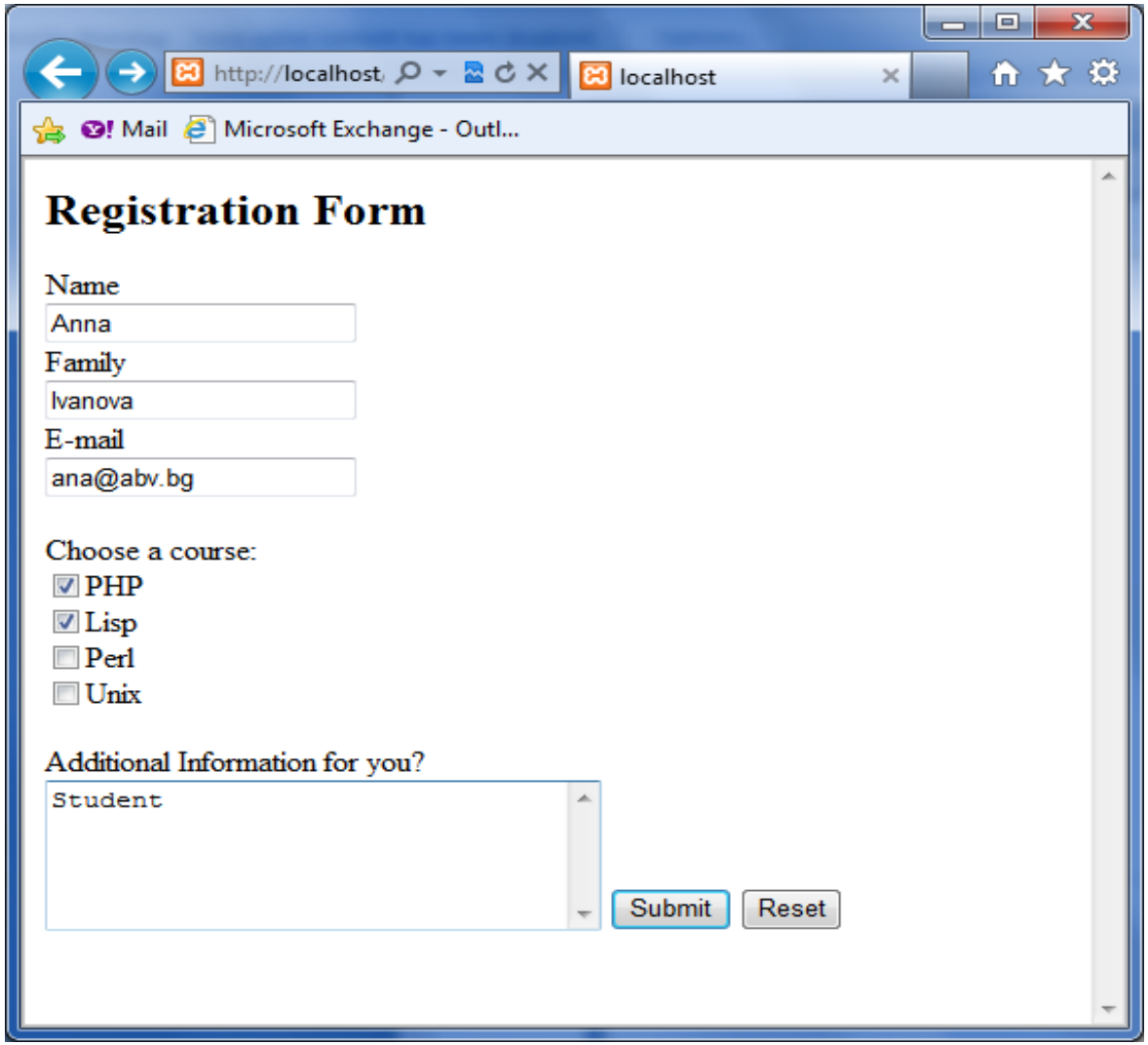
goto (Достъпен във версии 5.3.0 и по-нови)

- Оператор за безусловен преход към конкретна езикова конструкция.
- Мястото на прехода се дефинира с уникален идентификатор (label, етикет)
- Има архитектурни ограничения:
  - Управлението не може да се предава извън текущия файл
  - Управлението не може да се предава от/към функция или метод
  - Управлението не може да се предава към цикъл

```
<?php
goto label1;
...
label1:
echo “Hello world”
?>
```

- Идеята е да се попълни форма за регистрация на участници в курс по програмиране, като след регистрацията, участникът получава съобщение в прозореца на браузера…***Example222.html***

Демонстрационен пример 1



- След попълване на формата и натискане на Submit бутон, участникът получава следното съобщение в прозореца на браузера (***Скрипт 2.php, обработващ Example222.html***):

Здравейте, уважаеми , Anna Ivanova!

Съобщаваме ви, че Избраните от вас лекции ще бъдат на 12 май

- лекция по PHP в 14.30 (Вашият лектор е, Васил Василев)
- лекция по Lisp в 12.00 (Вашият лектор е, Иван Иванов)

С уважение, административен директор

```
<html>
<head>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;
charset=utf-8">
</head>
<body>
<h2> Registration Form </h2>
<form action="2.php" method=POST>
Name <br><input type=text name="first_name"  value=" Input your
name"><br>
Family <br><input type=text name="last_name"><br>
E-mail <br><input type=text name="email"><br>
<p> Choose a course:<br>
<input type=checkbox name='kurs[]' value='PHP'>PHP<br>
<input type=checkbox name='kurs[]' value='Lisp'>Lisp<br>
<input type=checkbox name='kurs[]' value='Perl'>Perl<br>
<input type=checkbox name='kurs[]' value='Unix'>Unix<br>
<P> Additional Information for you? <BR>
<textarea name="comment" cols=32 rows=5></textarea>
<input type=submit value="Submit">
<input type=reset value="Reset">
</form>
</body>
</html>
```

Example222.html

Скрипт 2.php, обработващ Example222.html

```
<?php
print '<META HTTP-EQUIV="Content-Type" CONTENT="text/html;
charset=utf-8">';
$times = array("PHP"=>"14.30","Lisp"=>"12.00",
"Perl"=>"15.00","Unix"=>"14.00");
$lectors = array("PHP"=>"Васил Василев", "Lisp"=>"Иван Иванов",
"Perl"=>"Петър Петров", "Unix"=>"Симеон Семов");
define("SIGN","С уважение, административен директор");
define("MEETING_TIME","18.00");
$date = "12 май";
$str = "Здравейте, уважаеми " . $_POST["first_name"] . " " .
$_POST["last_name"]."!<br>";
$str .= "<br>Съобщаваме ви, че ";
$kurses = $_POST["kurs"];
if (!isset($kurses))
{ $event = "Следващият курс в школата";
   $str .= "$event ще се състои на $date ". MEETING_TIME . "<br>";
} else {
$event = "Избраните от вас лекции ще бъдат на $date <ul>";
$lect = "";
for ($i=0;$i<count($kurses);$i++){
$k = $kurses[$i];
$lect = $lect . "<li>лекция по $k в $times[$k]";
$lect .= " (Вашият лектор е, $lectors[$k])";
}
$event = $event . $lect . "</ul>";
$str .= "$event";
}
$str .= "<br>". SIGN; // добавяме подпис
echo $str; // извеждаме съобщение на екрана
?>
```

Коментар на примера :

- Тук всичко е достатъчно понятно: единствено, интересен е начина на предаване на стойностите на елемент **checkbox**:
- Когато пишем **kurs[],** за име на елемента **checkbox**, това значи че първият чекнат елемент **checkbox** ще бъде записан в първия елемент на масива **kurs[]**, вторият - във втория и т.н. Може разбира се, да дадем различни имена на елементите от тип **checkbox**, но това ще усложни обработката на данните, ако курсовете са много на брой.
- Скриптът, който ще обработва **example222.html** е **2.php** (формата се обръща именно към него чрез атрибута **action**). По подразбиране, за предаване се използва метод **GET**, но ние указваме изрично POST.
- По получените сведения от регистриращ се човек, скриптът генерира съответстващо съобщение. Ако човекът е избрал някакви курсове, то му се изпраща съобщение за времето на провеждане на курса и за лекторите, които ги четат. Ако човекът нищо не е избрал, то се извежда някакво съобщение, например за следващия възможен курс към школата по програмиране.

Демонстрационен пример 2

- Искаме да създадем програма, която може да се използва за генерация на писма, с които се поканват множество клиенти, за участие в различни мероприятия.
- Целесъобразно е информацията за хората и събитията да е в отделен файл data.php. Можем да напишем програма, независеща (или може би малко зависеща) от тази информация и нейната структура. По този начин, ако се наложи да разширим списъка на адресантите, няма да е нужно да изменяме скрипта, генериращ поканата.Освен това, информацията за хората и събитията може да се използва и в други скриптове.
- В самия скрипт letters.php, генериращ поканата, ние използваме използване условни оператори, цикли, require и други изучени вече конструкции.

Решаване на една по сложна задача

- информацията за хората и събитията е в отделен файл **data.php**.

Пример 22. data.php

```
<?php
define("SIGN","С уважение, Деян Иванов");
// нашият подпис
// е константа

// информация за събитията
$events = array(
    "f" => "ден на отворените врати",
    "o" => "открито изложение",
    "p" => "бал на випускниците");
// налична информация за участниците
// (име и електронен адрес)
$people = array(
    "ivan" => array(
        "name" => "Иван Иванов",
        "email"=>"user_ivan@abv.bg"),
    "pit" => array(
        "name" => "Петър Петров",
        "email" => "user_petr@abv.bg"),
    "semen" => array(
        "name" => "Симеон Семов"));
// кой къде се поканва
$who_where["ivan"] = "o" ;
    // Иван - на изложение
$who_where["pit"] = "p";
    // Петър - на бал
$who_where["semen"] = "f";
    // Симеон - на ден на отворените врати
?>
```

Пример 23. letters.php

```php
<?php
require("data.php");
    // включваме файл с данни за събитията
foreach($people as $key => $man_info){
    // за всеки участник правим следното:
$event_key = $who_where[$key];
    // получаме събитието,
     // на което той се поканва
if ($event_key<>""){
    foreach($man_info as $key1 => $info){
    // получаваме име и email
     // на конкретния човек
    if ($key1=="name")
        $str = "Уважаеми (а), $info";
    if ($key1=="email") $email = $info;
     }  // съставяме поканата
$str .= "<br>Имаме удоволствието да ви поканим
на ".
         $events[$event_key];
    switch ($event_key){
     // в зависимост от събитието
    // добавяме някакво изречение
        case "f":
        $str .= "<br>Моля, потвърдете Вашето
              участие по телефона!";
        break;
        case "o":
        $str .= "<br>Бъдете 10
              минути преди откриването!";
        break;
        case "p":
    $str .= "<br>Не забравяйте да донесете
подарък :-)";
        break;
    }
    $str .= "<br>" . SIGN . "<hr>";
     // добавяме подпис
    echo $str; // извеждаме поканата на екрана

/* ако сте настроили изпращане на e-mail с PHP,
то писмото може и да се изпрати с команда:
mail($email,"Letter",$str); */
    }
}
?>
```

Пример 23. letters.php

Резултат:

```
Уважаеми (а), Иван Иванов
Имаме удоволствието да ви поканим на открито
изложение
Бъдете 10 минути преди откриването!
С уважение, Деян Иванов
```

```
Уважаеми (а), Петър Петров
Имаме удоволствието да ви поканим на бал на
випускниците
Не забравяйте да донесете подарък :-)
С уважение, Деян Иванов
```

```
Уважаеми (а), Симеон Семов
Имаме удоволствието да ви поканим на ден на
отворените врати
Моля, потвърдете Вашето участие по телефона!
С уважение, Деян Иванов
```

скрипта, генериращ поканата - letters.php.