PHP Web Development

```
require_once “settings.php”;
```

Модуль 2. Основы PHP. Управляющие конструкции

Автор: Владислав Бабак
linkedin.com/in/vladyslavbabak/

Управляющие конструкции

Управляющие конструкции позволяют управлять выполнением инструкций. При помощи управляющих конструкций можно выполнить инструкцию один или несколько раз, не выполнить вообще или выполнить частично.

Список управляющих конструкций: if, else, elseif/else if, while, do-while, for, foreach, break, continue, switch, declare, return, require, include, require_once, include_once, goto.

if/elseif/else if/else

Конструкция if является одной из наиболее важных во многих языках программирования, в том числе и PHP. Она предоставляет возможность условного выполнения фрагментов кода. Вычисляет выражение в булево значение. Если true - инструкция выполнится, если false - будет проигнорирована. Конструкции if могут быть бесконечно вложены в другие конструкции if, что дает большую гибкость в организации условного выполнения различных частей программы.

```
<?php

$name = ‘Mike’;
if (!empty($name)) {
    echo $mike;
}
```

if/elseif/else if/else

```php
<?php

$a = $_GET['a'] ?? '';
$b = $_GET['b'] ?? '';

if (strlen($a) > strlen($b)) {
    echo 'Длина a больше';
} else if (strlen($a) === strlen($b)) {
    echo 'Длина a равна b';
    if ($a === $b) {
        echo 'Строки равны';
    }
} else {
    echo 'Длина b больше';
}
```

switch

Конструкция switch подобна серии конструкций if. Ее удобно использовать если необходимо сравнить переменную или выражение с множеством разных значений. Конструкция switch/case использует неточное сравнение (==).

```
<?php
$a = mt_rand(0, 5);
switch ($a) {
     case “0”:
     case “1”:
     case “2”:
          echo $a . ‘ меньше 3‘;
          break;
     case “4”:
     case “5”:
          echo $a . ‘ больше 3’;
          break;
     default:
          echo $a . ‘ равно 3’;
          break;
}
```

declare

Конструкция declare используется для установки директив исполнения для блока кода. Синтаксис declare(directive);

В настоящее время распознаются только три директивы: директива ticks (тик - это событие, которое случается каждые N низкоуровневых операций, выполненных парсером внутри блока declare; значение N задается, используя ticks=N внутри секции directive блока declare; слушатель события регистрируется функцией register_tick_function()), директива encoding (кодировка скрипта может быть указана для каждого скрипта, например declare(encoding='ISO-8859-1');) и директива strict_types (строгая типизация).

```
<?php
declare(ticks=1);
function tick_handler()
{
    echo "tick_handler() выполнено\n";
}

register_tick_function('tick_handler');
```

declare - strict_types

По умолчанию, PHP будет пытаться привести значения несоответствующих типов к скалярному типу, если это возможно. Например, если в функцию передается integer, а тип аргумента объявлен string, в итоге функция получит преобразованное string значение.

Для отдельных файлов можно включать режим строгой типизации. В этом режиме в функцию можно передавать значения только тех типов, которые объявлены для аргументов. В противном случае будет выбрасываться исключение TypeError. Есть лишь одно исключение - integer можно передать в функцию, которая ожидает значение типа float. Вызовы функций внутри встроенных функций не будут затронуты директивой strict_types.

Режим строгой типизации распространяется на вызовы функций совершенные из файла, в котором этот режим включен, а не на функции, которые в этом файле объявлены. Строгая типизация применима только к скалярным типам.

Разберем на примере.

return

Конструкция return используется для возврата управления в программу из которой была вызвана. Выполнение программы продолжается с инструкции, следующей за местом вызова.

Если вызвано из функции, выражение return немедленно прекращает выполнение текущей функции и возвращает свой аргумент как значение данной функции. return также завершит выполнение выражения eval() или всего файла скрипта.

Если вызывается из глобальной области видимости, выполнение текущего файла скрипта прекращается. Если текущий файл скрипта был подключен с помощью функций include или require, тогда управление возвращается к файлу, который вызывал текущий. Более того, если текущий файл скрипта был подключен с помощью include, тогда значение переданное return будет возвращено в качестве значения вызова include. Если return вызывается из главного файла скрипта, тогда выполнение скрипта прекращается.

Разберем на примере.

require, include, require_once, include_once

Выражение include включает и выполняет указанный файл. Файлы включаются исходя из пути указанного файла, или, если путь не указан, используется путь, указанный в директиве include_path. Если файл не найден в include_path, include попытается проверить директорию, в которой находится текущий включающий скрипт и текущую рабочую директорию перед тем, как выдать ошибку. Конструкция include выдаст warning, если не сможет найти файл. Когда файл включается, его код наследует ту же область видимости переменных, что и строка, на которой произошло включение.

require идентично include за исключением того, что при ошибке выдает фатальную ошибку.

include_once аналог include но включает и выполняет указанный файл только один раз.

require_once аналог require но включает и выполняет указанный файл только один раз.

Разберем на примере.

goto

Выражение goto используется для перехода на метку кода. Целевая метка должна находиться в том же файле, в том же контексте. Имеется ввиду, что вы не можете ни перейти за границы функции или метода, ни перейти внутрь одной из них.

Вы также не можете перейти внутрь любой циклической структуры или оператора switch. Но вы можете выйти из них, и обычным применением оператора goto является использование его вместо многоуровневых break.

Разберем на примере.

```
<?php

goto skip_code;

echo 'This code will be skipped';

skip_code:
echo 'End of file';
```

while

Циклы while являются простейшим видом циклов в PHP. Указывает PHP выполнять вложенные выражения повторно до тех пор, пока выражение в самом while является TRUE.

```php
<?php

// example 1
$i = 1;
while ($i <= 10) {
    echo $i++;
}

// example 2
$j = 1;
while ($i > 0 && $j < 10) {
     echo "i " . $i-- . PHP_EOL;
     echo "j " . $j++ . PHP_EOL;
}
```

do-while

Цикл do-while очень похож на цикл while, с тем отличием, что истинность выражения проверяется в конце итерации, а не в начале. Первая итерация цикла всегда выполняется.

Для выхода из цикла используется break; а для того чтобы пропустить текущую итерацию - continue;

```php
<?php

// example 1
$i = 0;
do {
    echo $i;
} while ($i > 0);
```

for

Цикл for - один из видов циклов, имеет вид for (expr1; expr2; expr3) {}

Первое выражение (expr1) всегда выполняется только один раз в начале цикла.

В начале каждой итерации оценивается выражение expr2. Если оно принимает значение TRUE, то цикл продолжается, и блок инструкций будет выполнен. Если оно принимает значение FALSE, выполнение цикла заканчивается.

В конце каждой итерации выполняется выражение expr3.

Каждое из выражений может быть пустым или содержать несколько выражений, разделенных запятыми.

```
<?php

// example 1
for ($i = 0; $i < 10; $i++) {
     echo $i;
}

// example 2
for ($i = 8, $j = 5; $i < 10;) {
    echo ++$i;
}
echo $i;
```

foreach

Конструкция foreach предоставляет простой способ перебора массивов.

Foreach работает только с массивами и объектами, и будет генерировать ошибку при попытке использования с переменными других типов или неинициализированными переменными.

Пример записи: foreach ($array as $key => $value)

Часть $key => можно опустить.

```php
<?php

// example 1
$arr = array(1, 2, 3, 4);
foreach ($arr as &$value) {
    $value = $value * 2;
}
unset($value);
```

break

Конструкция break прерывает выполнение текущей структуры for, foreach, while, do-while или switch.

break принимает необязательный числовой аргумент (больше 0 и меньше уровня вложенности цикла, иначе будет сгенерирована ошибка fatal error), который сообщает ему выполнение какого количества вложенных структур необходимо прервать. Значение по умолчанию 1, только ближайшая структура будет прервана.

continue

Конструкция continue используется внутри циклических структур для пропуска оставшейся части текущей итерации цикла и, при соблюдении условий, начала следующей итерации.

continue принимает необязательный числовой аргумент, который указывает на скольких уровнях вложенных циклов будет пропущена оставшаяся часть итерации. Значением по умолчанию является 1, при которой пропускается оставшаяся часть текущего цикла. Аналогично с break генерирует fatal error если числовой аргумент меньше 1 или больше числа вложенных структур.

Альтернативный синтаксис

Для switch, if, for, while, foreach возможен альтернативный синтаксис. В каждом случае основной формой альтернативного синтаксиса является изменение открывающей фигурной скобки на двоеточие (:), а закрывающей скобки на endif;, endwhile;, endfor;, endforeach; или endswitch; соответственно. Смешивание синтаксиса в одном и том же блоке управления не поддерживается.

Любой вывод (включая пробельные символы) между выражением switch и первым case приведут к синтаксической ошибке.

```
<div>
<?php if ($a == 5): ?>
A равно 5
<?php endif; ?>
</div>
```

Другие конструкции PHP

echo - Выводит строку.

print - Выводит строку. Всегда возвращает int(1), не является оператором.

Конструкции echo/print могут использоваться как с круглыми скобками так и без.

exit, die - Выводит сообщение и прекращает выполнение текущего скрипта.

```
<?php
echo 'Hi,', ' ', 'Frank', PHP_EOL;
echo ('Hi. ' . 'How are you?'); // внутри скобок нужно использовать конкатенацию

echo print (22); // 221
```

Спасибо за внимание!

- остались вопросы?
- подведение итогов
- переход к практике
- домашнее задание

IN: linkedin.com/in/vladyslavbabak
Skype: marmalade.vlad
E-Mail: marmalade.vlad@gmail.com