Тема 3.4
«Управляющие конструкции PHP»

Вступление

PHP поддерживает «классический» набор условных конструкций: ветвления (if), выбор вариантов (switch), циклы.

Итак…

Оператор условия if

Простейшей условной конструкцией является оператор ветвления if, который может дополняться ветками альтернативного ветвления elseif и else.

```
if ($a==10) echo “OK”;
if ($b==20) echo “Yes”; else echo “No”;
```

Несмотря на то, что в случае, когда за условием следует один оператор, для удобства чтения рекомендуется всё равно использовать т.н. «операторные скобки» { } (их смысл, как следует из названия – представить набор операторов как «один большой оператор»).

```
if ($a==10)
 {
  echo “OK”;
}
```

Оператор условия if

Альтернативное ветвление elseif позволяет задавать сложную логику выбора вариантов:

```
if ($a==10)
{
 echo “A”;
}
 elseif ($A==20)
  {
   echo “B”;
  }
   elseif ($a==30)
    {
     echo “C”;
    }
     // и т.д.
     else
      {
       echo “Undefined”;
      }
```

Оператор условия if

Условия могут быть вложенными:

```
if ($a==10)
 {
  if ($z=“OK”)
   {
     echo “Super!”
   }
    else
   {
     echo “Normal”;
   }
 }
  else
{
  echo “Undefined”;
}
```

Оператор условия if

Условия могут быть сложными (составными), здесь пригодятся уже рассмотренные нами ранее логические операторы:

```
if (($a==10)&($b==20)) …
if (($a!=5)||($c>12)||($str==OK“”)) …
```

Обратите внимание: в круглые скобки берётся как всё сложное условие целиком, так и каждое простое условие в его составе по отдельности:

```
if ((($a!=5)||($c>12)||($str==OK“”)) …
```

Оператор условия if

Общий синтаксис условного оператора if таков:

if (condition) statement_1 else statement_2

Условие condition может быть любым выражением (в т.ч., например, вызовом функции). Если оно истинно, то выполняется оператор statement_1. В противном случае выполняется оператор statement_2. Допустима сокращенная форма записи условного оператора, в которой отсутствуют else и оператор statement_2.

В свою очередь, операторы statement_1 и statement_2 могут быть условными, что позволяет организовывать цепочки проверок любой глубины вложенности. И в этих цепочках каждый условный оператор может быть как полным, так и сокращенным.

В связи с этим возможны ошибки неоднозначного сопоставления if и else. Синтаксис языка предполагает, что при вложенных условных операторах каждое else соответствует ближайшему if.

Использование альтернативного ветвления elseif достаточно распространено, однако рекомендуется в целях повышения читаемости кода заменять такие громоздкие конструкции значительно более наглядными с использованием оператора switch, который мы сейчас и рассмотрим.

Переключатель switch

Переключатель switch является наиболее удобным средством для организации т.н. «мультиветвления». Синтаксис переключателя switch таков:

```
switch(expression) // переключающее выражение
{
   case value1: // константное выражение 1
      statements; // блок операторов 1
   break;
   case value2: // константное выражение 2
      statements; // блок операторов 2
   break;
   default:
      statements; // блок операторов N
}
```

Переключатель switch

Переключатель switch передаёт управление тому из помеченных case операторов, для которого значение константного выражения совпадает со значением переключающего выражения. Если значение переключающего выражения не совпадает ни с одним из константных выражений, то выполняется переход к оператору, помеченному меткой default (если он есть). В переключателе switch может быть не более одной метки default, однако она может отсутствовать вообще.

Так же, как и в случае условного оператора if для переключателей допустимы любые степени вложенности, однако злоупотреблять этим не следует.

Выход из переключателя осуществляется с помощью оператора break.

В PHP в качестве меток case могут использоваться не только литералы: сами метки могут быть переменными. В качестве меток case в PHP не могут выступать только массивы и объекты.

Рассмотрим пример:

Переключатель switch

```
switch ($a)
{
 case 10:
   echo “Ten”;
   echo “OK”;
   break;
case “ZZZ”:
   echo “Some string”;
   echo “OK”;
   break;
default:
   echo “Something unknown”;
}
```

Переключатель switch

Обратите внимание: оператор break ОБЯЗАТЕЛЬНО должен присутствовать в конце каждого case, т.к. выполнение кода происходит «от первого совпадения в case и до первого следующего break или (если break нет) до конца switch», что, фактически, может привести к выполнению блоков операторов, относящихся к другим условиям, прописанным в case.

Циклы

Операторы цикла задают многократное исполнение операторов в теле цикла. В PHP определены 4 разных оператора цикла:

1. Цикл с предусловием while:

```
while (condition)
{
  statements;
}
```

2. Цикл с постусловием do…while:

```
do
{
  statements;
} while (condition);
```

3. Итерационный цикл for:

```
for (expression1;expression2;expression3)
{
  statements;
}
```

4. Итерационный цикл foreach:

```
foreach (array as [$arr_key =>] $arr_value)
{
  statements;
}
```

Если три первых оператора цикла берут свое начало от C-подобных языков, то последний оператор позаимствован у языка Perl. Цикл foreach мы рассмотрим отдельно в теме, посвящённой массивам в PHP.

Цикл с предусловием – while

Оператор while называется оператором цикла с предусловием. При входе в цикл вычисляется выражение условие, и, если его значение истинно, выполняется тело цикла. Затем вычисления выражения условия и операторов тела цикла выполняется до тех пор, пока значение выражения условия не станет ложным.

Оператором while удобно пользоваться для просмотра всевозможных последовательностей, если в конце них находится заранее известный символ.

Пример простейшего цикла while:

```
$var = 5;
$i = 0;
while(++$i <= $var)
 {
   echo($i).”<br />”;
 }
```

Цикл с предусловием – while

Для «досрочного» выхода из цикла применяется оператор break. При обнаружении этого оператора текущая итерация цикла прекращается, и последующие итерации не происходят.

Иногда бывает нужно прервать только текущую итерацию, и перейти сразу к следующей. Для этого применяется оператор continue, который позволяет пропустить все операторы до конца данной итерации и сразу начать новую итерацию.

Бесконечный цикл реализуется при помощи оператора while следующим образом:

```
while(TRUE)
{
    ...
}
```

Цикл с постусловием – do … while

Этот набор операторов называется циклом с постусловием. При входе в цикл в любом случае выполняется тело цикла (т.е. **цикл всегда будет выполнен хотя бы один раз**), затем вычисляется условие, и если оно не равно FALSE, вновь выполняется тело цикла.

В нижеследующем примере ноль всегда будет добавлен в список, независимо от условия:

```
$var = 5;
$i = 0;
do
 {
  echo($i).”<br />”;
 }
while (++$i <= $var)
```

Цикл с постусловием бывает полезен при обработке таких последовательностей, обработку которых нужно заканчивать не до, а после появления «концевого признака». Бесконечный цикл реализуется так:

```
do ; while(TRUE);
```

Итерационный цикл – for

Итерационный цикл for имеет следующий формат:

```
for(expression_1; expression_2; expression_3)
{
   statements;
}
```

Здесь expression_1 (инициализация цикла) – последовательность определений и выражений, разделяемая запятыми. Все выражения, входящие в инициализацию, вычисляются **только один раз при входе в цикл**.

Как правило, здесь устанавливаются начальные значения счётчиков и параметров цикла. Смысл выражения-условия expression_2 такой же как и у циклов с пред- и постусловиями.

При отсутствии выражения-условия предполагается, что его значение всегда истинно.

Выражения expression_3 (разделяются запятыми) вычисляются **в конце каждой итерации** после выполнения тела цикла.

Бесконечный цикл можно организовать следующим образом:

for(;;); или for(;TRUE;);

Итерационный цикл – for

Пример итерационного цикла:

```
for ($i=0;$i<100;$i++)
{
 echo $i.”<br />”;
}
```

Тернарный оператор

Для закрепления вы памяти вспомним, что управление условием выполнения программы можно реализовывать и с помощью тернарного оператора – сокращённого аналога оператора if:

```
// Пример использования тернарного оператора
$action = (empty($_POST['action'])) ? 'default' : $_POST['action'];

// Приведённый выше код аналогичен следующему
if (empty($_POST['action']))
{
    $action = 'default';
}
else
{
    $action = $_POST['action'];
}
```