### Лекция 11.

### Тема. Операторы включения

#### include

Оператор *include* позволяет включать код, содержащийся в указанном файле, и выполнять его столько раз, сколько программа встречает этот *оператор*. Включение может производиться любым из перечисленных способов:

include 'имя\_файла';

include $file\_name;

include ("имя\_файла");

**Пример 3.9**. Пусть в файле params.inc у нас хранится набор каких-то параметров и функций. Каждый раз, когда нам нужно будет использовать эти параметры (функции), мы будем вставлять в текст нашей основной программы команду include ('params.inc').

params.inc

<?php

$user = "Вася";

$today = date("d.m.y");

/\* функция date() возвращает дату

и время (здесь – дату в формате

день.месяц.год) \*/

?>

include.php

<?php

include ("params.inc");

/\* переменные $user и $today заданы в файле

params.inc. Здесь мы тоже можем ими

пользоваться благодаря команде

include("params.inc") \*/

echo "Привет, $user!<br>";

// выведет "Привет, Вася!"

echo "Сегодня $today";

// выведет, например, "Сегодня 7.07.05"

?>

Пример 3.9. Использование оператора включения include

Заметим, что использование оператора *include* эквивалентно простой вставке содержательной части файла params.inc в код программы include.php. Может быть, тогда можно было в params.inc записать простой текст без всяких тегов, указывающих на то, что это php-код? Нельзя! Дело в том, что в момент вставки файла происходит переключение из режима обработки PHP в режим HTML. Поэтому код внутри включаемого файла, который нужно обработать как *PHP-скрипт*, должен быть заключен в соответствующие теги.

Поиск файла для вставки происходит по следующим правилам.

1. Сначала ведется поиск файла в *include\_path* относительно текущей рабочей директории.
2. Если файл не найден, то поиск производится в *include\_path* относительно директории текущего скрипта.
3. Параметр *include\_path* , определяемый в *файле настроек* PHP, задает имена директорий, в которых нужно искать *включаемые файлы*.

Например, ваш *include\_path* это . (то есть *текущая директория*), текущая рабочая директория это /www/. В основной файл include.php вы включаете файл my\_dir/a.php, который в свою очередь включает b.php. Тогда *парсер* первым делом ищет файл b.php в директории /www/, и если такового нет, то в директории /www/my\_dir/.

Если файл включен с помощью *include* , то содержащийся в нем код наследует область видимости переменных строки, где появился *include* . Любые переменные вызванного файла будут доступны в вызывающем файле с этой строки и далее. Соответственно, если *include* появляется внутри функции вызывающего файла, то код, содержащийся в вызываемом файле, будет вести себя так, как будто он был определен внутри функции. Таким образом, он унаследует область видимости этой функции. Хотя мы и не знакомились еще с понятием функции, все же приводим здесь эти сведения в расчете на интуитивное его понимание.

**Пример 3.10**. Пусть файл для вставки params.inc останется таким же, а include.php будет следующим:

<?php

function Footer(){

// объявляем функцию с именем Footer

include ("params.inc");

/\* включаем файл params.inc.

Теперь его переменными можно пользоваться,

но только внутри функции \*/

$str = "Сегодня: $today <br>";

$str .= "<a

href='mailto:help@intuit.ru'>Страницу

создал $user</a>";

echo "$str";

}

Footer();

// вызываем функцию Footer(). Получим:

//Сегодня: 08.07.05

//Страницу создал Вася

echo "$user, $today";

// выведет запятую, так как

// эти переменные видны только

// внутри функции

?>

Пример 3.10. Область видимости при использовани include

Кроме локальных файлов, с помощью *include* можно включать и внешние файлы, указывая их url-адреса. Данная возможность контролируется директивой url\_fopen\_wrappers в *файле настроек* PHP и по умолчанию, как правило, включена. Но в версиях PHP для Windows до PHP 4.3.0 эта возможность не поддерживается совсем, вне зависимости от url\_fopen\_wrappers.

include() – это специальная языковая конструкция, поэтому при использовании внутри условных блоков ее нужно заключать в фигурные скобки.

<?php

/\* Это неверная запись. Получим ошибку.

Мы же вставляем не одну команду,

а несколько, они только записаны

в другом файле \*/

if ($condition) include("first.php");

else include("second.php");

// А вот так правильно.

if ($condition){ include("first.php"); }

else { include("second.php"); }

?>

Пример 3.11. Использование include()

При использовании *include* возможны два вида ошибок – ошибка вставки (например, нельзя найти указанный файл, неверно написана сама команда вставки и т.п.) или ошибка исполнения (если ошибка содержится во вставляемом файле). В любом случае при ошибке в команде *include* исполнение скрипта не завершается.

#### require

Этот оператор действует примерно так же, как и #include в C++. Все, что мы говорили о *include* , лишь за некоторыми исключениями, справедливо и для *require* . *require* также позволяет включать в программу и исполнять какой-либо файл. Основное отличие *require* и *include* заключается в том, как они реагируют на возникновение ошибки. Как уже говорилось, *include* выдает предупреждение, и работа скрипта продолжается. Ошибка в *require* вызывает фатальную ошибку работы скрипта и прекращает его выполнение.

*Условные операторы* на require() не влияют. Хотя, если строка, в которой появляется этот оператор, не исполняется, то ни одна строка кода из вставляемого файла тоже не исполняется. Циклы также не влияют на require(). Хотя код, содержащийся во вставляемом файле, является объектом цикла, но вставка сама по себе происходит только однажды.

В реализациях PHP до версии 4.0.2 использование require() означало, что интерпретатор обязательно попытается прочесть вставляемый файл.

*require* , как и *include* , при использовании внутри условных блоков нужно заключать в фигурные скобки.

### Решение задачи

И наконец, вернемся к задаче, сформулированной в начале лекции. Мы хотим создать программу, которую можно было бы использовать для отправки писем (или просто для их генерации) с приглашениями на различные мероприятия множеству пользователей. В предыдущей лекции уже рассматривался подобный случай. Сейчас мы вынесем всю информацию о людях и событиях в отдельный *файл* data.php и напишем программу, не зависящую (ну, может, совсем чуть-чуть зависящую) от этой информации и ее структуры. В этом случае для того, чтобы, например, расширить *список* адресатов, не нужно будет изменять *скрипт*, генерирующий приглашения. Кроме того, можно будет использовать информацию о людях и событиях в других скриптах. В самом скрипте, генерирующем приглашения letters.php, мы использовали *условные операторы*, циклы, *require* и другие изученные ранее конструкции.

<?php

define("SIGN","С уважением, Вася");

// пусть наша подпись

// будет константой

// информация о событиях

$events = array(

"f" => "день открытых дверей",

"o" => "открытие выставки",

"p" => "бал выпускников");

// имеющаяся информация о людях

// (имя и электронный адрес)

$people = array(

"ivan" => array(

"name" => "Иван Иванович",

"email"=>"user\_ivan@intuit.ru"),

"pit" => array(

"name" => "Петр Петрович",

"email" => "user\_petr@intuit.ru"),

"semen" => array(

"name" => "Семен Семенович"));

// кто куда приглашается

$who\_where["ivan"] = "o" ;

// Иван – на выставку

$who\_where["pit"] = "p";

// Петр – на бал

$who\_where["semen"] = "f";

// Семен – на день открытых дверей

?>

Листинг 3.12. data.php

<?php

require("data.php");

// включаем файл с данными о событиях

foreach($people as $key => $man\_info){

// для каждого человека делаем следующее:

$event\_key = $who\_where[$key];

// получаем событие,

// на которое он приглашается

if ($event\_key<>""){

foreach($man\_info as $key1 => $info){

// получаем имя и email

// конкретного человека

if ($key1=="name")

$str = "Уважаемый (ая), $info";

if ($key1=="email") $email = $info;

}

// составляем приглашение

$str .= "<br>Приглашаем Вас на ".

$events[$event\_key];

switch ($event\_key){

// в зависимости от события

// добавляем какую-нибудь строчку

case "f":

$str .= "<br>Подтвердите Ваше

участие по телефону!";

break;

case "o":

$str .= "<br>Приходите за 15

минут до открытия!";

break;

case "p":

$str .= "<br>Не забудьте подарок :-)";

break;

}

$str .= "<br>" . SIGN . "<hr>";

// добавляем подпись

echo $str; // выводим приглашение на экран

/\* если у вас настроена отправка почты с

помощью PHP, то письмо можно отправить

командой mail($email,"Letter",$str); \*/

}

}

?>

Листинг 3.13. letters.php

### Заключение

Итак, мы завершили изучение *управляющих конструкций* языка *PHP*. Мы старались привести самое основное и не углубляться в различные нюансы тех или иных конструкций. Подробную информацию можно найти в руководстве по *PHP*.