PART9

- 流程控制

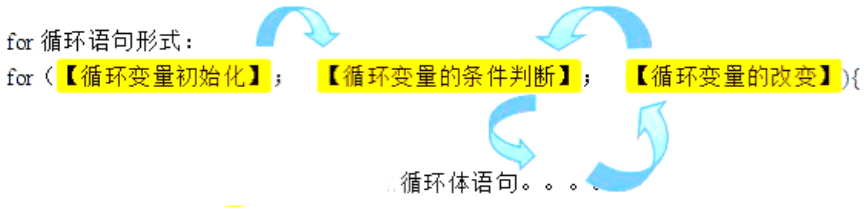
循环结构(循环的中断）
循环中，有两种中断语句可以使用：
break：用于完全终止某个循环，让执行流程进入到循环语句后面的语句；
continue：用于停止当前正在进行的当次循环，而进入到循环的“下一次”过程中去（通常就是循环　　的开始位置）；

在php中，该两个循环有更强的能力：中断“更多层”的循环，语法如下：
break 正整数n;　　//比如1,2，3；
continue 正整数n;　　//比如1,2，3；

循环的“层”，指的是从当前中断语句（break或continue）算起，往代码的“外部”数循环的个数，就是层数。比如：

```
for(...){       //循环1
    for(...){       //循环2
        for(...){       //循环3
            break 2; //此时会中断循环2：其实指中断“2层”
            //对此break语句，循环3是其“第一层”，循环2是其第2层，循环1是其第3层
}
        contine 2;    //此时会中断循环12：其实指中断“2层”
            //对此contine语句，循环2是其“第1层”，循环1是其第2层，
    }
}
```

- for循环语句



- while循环语句：

```
【循环变量初始化】
while(【循环变量的条件判断】){

//循环体语句。。。
【循环变量的改变】
}
```

- do while循环语句：

```
【循环变量初始化】
do {

//循环体语句。。。
【循环变量的改变】
}while(【循环变量的条件判断】)；
```

说明：
1，do while会先进入循环体执行一次（不判断条件）；
2，然后，判断循环条件是否满足，如果满足，又会回到do的开始位置（进入循环体）执行——这就是循环的正常情况。
3，如果不满足，就结束循环。

- 流程控制的替代语法

if(...)：。。。。 endif;

if (...) : 。。。else: 。。。 endif;
if (...): 。。。elseif(...) : 。。。 elseif(...): 。。。 else: 。。。 endif;
switch(...) : case ... case ... endSwitch;
while(...): 。。。endwhile;
for(...; ...; ...): 。。。 endfor;

```php
<?php
header('Content-type:text/html; charset=utf-8');


$v1 = 5;
if($v1 > 0):
    echo "abcd";
    echo "132";
else:
    echo "XYZ";
endif;
   ?>
```

- goto语句：

学此语句的目的是：不要用它！
语法形式：
//程序从这里开始：

标识符2：
语句群1.。。。。。

goto 标识符1;　含义是：立即跳转到标识符1所在位置的下一行继续执行
语句群2.。。。。。

标识符1:
语句群3.。。。。。

goto 标识符2:；

- 控制脚本执行顺序语句

die(字符串)/exit(字符串)：输出该字符串后，立即停止php的执行！即后续程序不再执行，包括后续的其他所有php和html代码部分。

exit是die的同义词。他们也可以不加字符串，而是直接停止。

```php
13  echo "<br/>";
14  echo "<hr/>";
15
16  echo "A";
17  echo "25";
18  die("<br/>baybay");
19  echo "A";
20  echo "25";
21     ?>
```

A25
baybay

sleep ($n)
让程序停止运行指定的秒数。然后等待过了那个时间后，就继续运行！

注意，其单位是“秒”；

```php

  echo "A";
  echo "<br/>当前时间" . date("Y-m-d H:i:s");
  echo "<br/>wait";
  sleep(3);
  echo "<br/>当前时间" . date("Y-m-d H:i:s");
  echo "<br/>25";
  die("<br/>baybay");
  echo "A";
  echo "25";
    ?>
```

```
A
当前时间2017-01-18 22:34:57
wait
当前时间2017-01-18 22:35:00
25
baybay
```

- 文件加载

<u>综述和基本语法：</u>
1，有4个文件加载语句：include, require, include_once, require_once
2，他们的使用形式完全一样，比如： include “要加载的文件路径”；或： include（“要加载的文件路径”）;
3，他们的含义也几乎完全一样：只是在加载失败时或是否重复加载这种情况，有所不同。
4，他们可以载入php或html文件；

<u>文件加载的路径问题：</u>
前提说明：以下的说明举例，以include为例，也适用于其他3个加载语句；
有3中路径形式可以使用：
相对路径：./ ../
是相对于当前网页文件所在的位置来定位某个被加载的文件位置，主要依赖以下2个特殊的路径符号：
./ ：表示当前位置，即当前网页文件所在的位置（目录）；
../ ：表示上一级位置，即当前网页文件所在的位置的上一级位置（目录）；
我们需要用这2个符号来表达位置信息，比如：
include ‘./page1.php’； 表示当前网页文件所在位置的page1.php文件；
include ‘../page2.php’；
include ‘../ab/page3.html’；

绝对路径：分2种：
本地绝对路径：
如：
include “c:/d1/d2/p1.php”；
include “d:/wamp/php/p2.html”；
<u>特别注意：我们其实几乎都不应该在代码中直接写这种本地绝对路径！</u>
但，其实我们这种本地绝对路径的写法是很常用的！
示例：

```php
<?php
header('Content-type:text/html; charset=utf-8');


echo "<br/>使用相对路径";
include './liucheng.php';

echo "<br/>使用绝对路径1";
include __DIR__ . '\liucheng.php';

echo "<br/>使用绝对路径2";
$root = $_SERVER['DOCUMENT_ROOT'];
include  $root ."\liucheng.php";

?>
```

```
使用相对路径abcd132
--------------------------------------------------------------------------------
A
当前时间2017-01-18 22:52:50
wait
当前时间2017-01-18 22:52:50
25A25
使用绝对路径1abcd132
--------------------------------------------------------------------------------
A
当前时间2017-01-18 22:52:50
wait
当前时间2017-01-18 22:52:50
25A25
使用绝对路径2abcd132
--------------------------------------------------------------------------------
A
当前时间2017-01-18 22:52:50
wait
当前时间2017-01-18 22:52:50
25A25
```

网络绝对路径：

比如：

include “http://www.abc.com/p1.php”；

include “http://www.baidu.com/index.php”；

- “无路径”（不推荐）：

形式就是没有给出路径信息，而只给出文件名，我们不推荐。

比如：include ‘page1.php’； //此时通常其实php语言引擎会在当前网页目录下找该文件。

- 文件载入和执行过程详解

第1步：从include语句处退出php脚本模式（进入html代码模式）

第2步：载入include语句所设定的文件中的代码，并执行（如同在当前文件中一样）

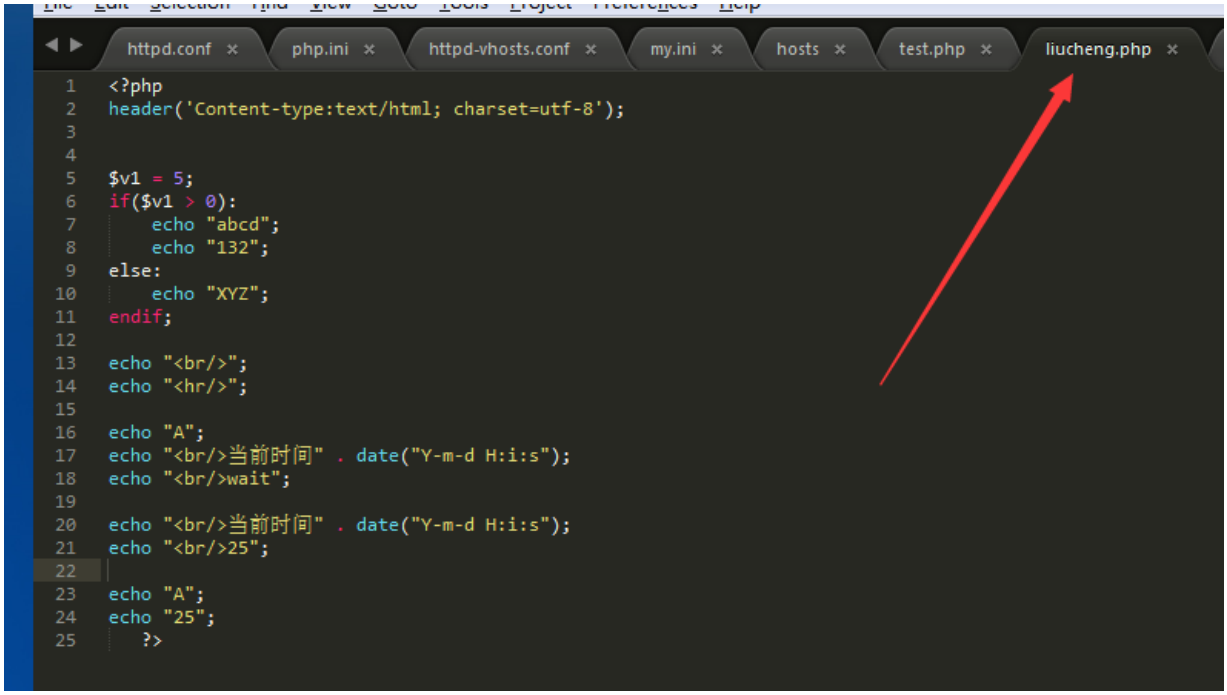
第3步：退出html模式重新进入php脚本模式，继续执行之后的代码

示例：

```
主文件开始位置:
<?php
header('Content-type:text/html; charset=utf-8');


    echo "<br />主文件中的位置A";
    include "./liucheng.php";   //要载入的文件
    echo "<br />主文件中的位置B";


?>
<br />主文件结束位置:
```




```
主文件开始位置:
主文件中的位置Aabcd132
---
A
当前时间2017-01-18 23:00:08
wait
当前时间2017-01-18 23:00:08
25A25
主文件中的位置B
主文件结束位置:
```

- 4个载入语句的区别

include和require的区别：
include载入文件失败时（即没有找到该文件），报一个“提示错误”，然后继续执行后续代码；
require载入文件失败时，报错并立即终止执行。
通常，require用于在程序中，后续的代码依赖于载入的文件的时候。

include_once和require_once的区别：
同include和require的区别：

include和include_once的区别：
include载入的文件不判断是否重复，只要有include语句，就会载入一次——即此时可能导致重复载入。

include_once载入的文件会有内部判断机制是否“前面代码”已经载入过，如果载入过，就不再载入。

require和require_once的区别：
同include和include_once的区别。