

文件加载:

基本语法: include, require, include_once, require_once

文件加载:

绝对路径

```
echo "<p>使用相对路径载入";  
include './page1.php';  
  
echo "<p>使用绝对路径载入(方法1)";  
include __DIR__ . '\page1.php';  
  
echo "<p>使用绝对路径载入(方法2)";  
$root = $_SERVER['DOCUMENT_ROOT']; //获得当前站点的跟目录  
include $root . "\day5" . '\page1.php';
```

使用相对路径载入
这是被载入页面page1.php

使用绝对路径载入(方法1)
这是被载入页面page1.php

使用绝对路径载入(方法2)
这是被载入页面page1.php

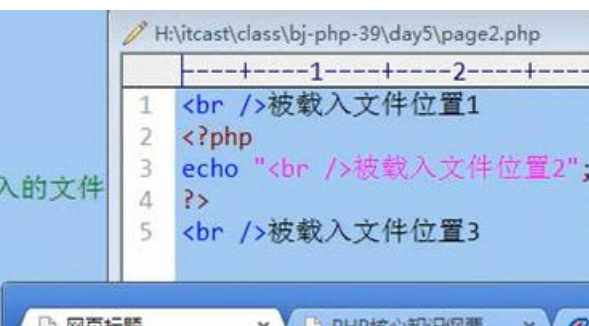
文件载入和执行过程:

第1步: 从 include 语句退出 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, 并执行之 (如同在当前文件中一样)

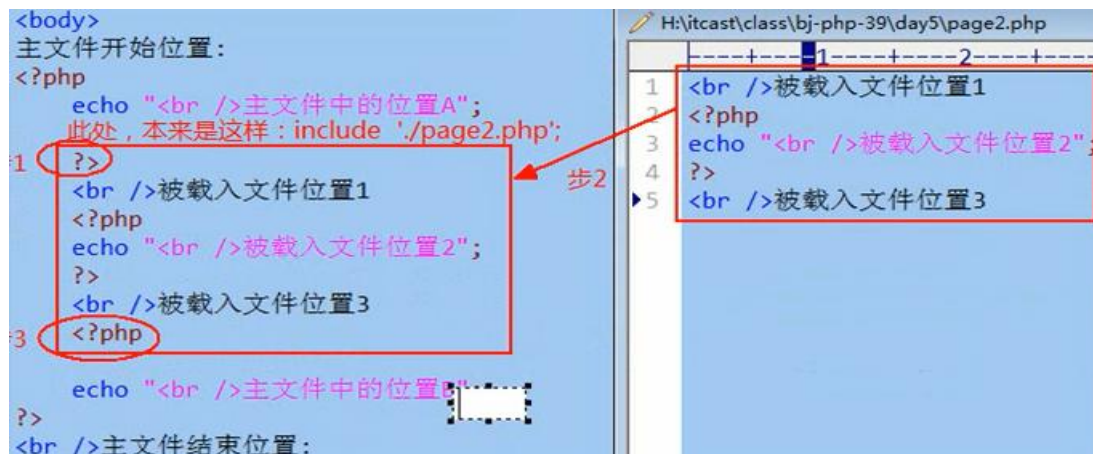
第3步: 退出 html 模式重新进入 php 脚本模式, 继续之后的代码。

```
<body>  
主文件开始位置:  
<?php  
    echo "<br />主文件中的位置A";  
  
    include "./page2.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:
```



相当于:

```
<body>  
主文件开始位置:  
<?php  
    echo "<br />主文件中的位置A";  
    此处, 本来是这样: include './page2.php';  
1  ?>  
    <br />被载入文件位置1  
    <?php  
    echo "<br />被载入文件位置2";  
    ?>  
    <br />被载入文件位置3  
3  <?php  
    echo "<br />主文件中的位置B";  
    ?>  
<br />主文件结束位置:
```



4 个载入语句的区别:

require, 和 include 的区别:

引用失败 (出错) 时, include 警告并继续, require 终止。

通常, require 用于在程序中, 后续代码依赖于载入的文件的时候。

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(可能导致重复载入)

include_once 载入的文件有内部判断机制是否“前面代码”已经载入,加载一次不重复加载

Require 和 require_once 的区别:

同 include 和 include_once 的区别。

在被载入的文件中 return 语句的作用:

1.一个载入语句是有返回值的,如果载入成功返回 1,如果载入失败返回 false

```
</head>
<body>
<?php
include "./page2.php";//成功就有返回值1,但这里不用它
$v1 = include "./page2.php";//有返回值,并放入变量v1
echo "<br />"; var_dump($v1);
$v2 = include "./no_this_page.php";//有返回值,是false
echo "<br />"; var_dump($v2);
?>
</body>
</html>
```

被载入文件位置1
被载入文件位置2
被载入文件位置3
被载入文件位置1
被载入文件位置2
被载入文件位置3
int(1)
Warning: include(./no_this_page.php) [function.include] failed to open stream: No such file or directory in H:\itcast\class\bj-php-39\da...
Warning: include() [function.include] failed to open stream: No such file or directory in H:\itcast\class\bj-php-39\da...
bool(false)

但,如果被载入的文件中有 return 语句,此时就有另外的机制和作用

2.return 语句的作用是终止载入过程--该 return 语句的后续载入文件的代码不再载入。

```
echo "<hr />";
$v3 = include_once "./page3.php";//此被载入文件中有return
echo "<br />111";
?>
```

```
1 <?php
2 echo "<br />被载入文件: aaa";
3 return ;
4 echo "<br />被载入文件: bbb";
5 ?>
```

运行结果为:

```
被载入文件: aaa
111
```

3.return 语句也可以用于被载入文件时返回一个数据,形式为: return xx 数据

```
echo "<hr />";
$v4 = include_once "./page4.php";//此被载入文件中有return
echo "<br />111";
echo "<br />"; var_dump($v4);
?>
```

```
1 <?php
2 $v1 = 123;
3 echo "<br />被载入文件: aaa";
4 return $v1; //此时会返回数据$v1
5 echo "<br />被载入文件: bbb";
6 ?>
```

运行结果:

```
被载入文件: aaa
111
int(123)
```

错误处理

错误处理的分类

语法错误: 程序运行之前都要先检查语法,如果语法有错误,就立即报错并不会执行程序

运行时错误：就是在语法检查通过后，开始运行程序并在此过程中遇到的错误，常见有 3 中：

提示性错误 警告性错误 致命性错误：

逻辑错误：指的是，程序本身可以执行，但计算结果错误

错误的分级：

php 语言中，对错误进行了分类归纳，每一级错误都有一个“代号（系统常量）”

系统常见错误：

E_ERROR: 致命错误

E_WARNING: 警告错误

E_NOTICE: 提示性错误

用户自定义错误：

E_USER_ERROR: 自定义致命错误

E_USER_WARNING: 自定义警告错误

E_USER_NOTICE: 自定义提示性错误

其他：

E_STRICT 严谨性语法检查错误

E_ALL 代表“所有错误”

错误代码的实际值：

```
function GetBinStr($e){
    $s = decbin($e); //这是一个二进制数字字符串
    //str_pad($str1, 长度n, $str2, 位置w)函数的作用是：
    //将字符串$str1,用字符串$str2填充到指定的长度n,
    //而且可以指定填充的位置w: 左边填充还是右边填充
    $s1 = str_pad($s, 16, "0", STR_PAD_LEFT);
    return $s1;
}
echo "<pre>";
echo "<br />E_ERROR=" . E_ERROR . ", \t\t其对应二进制值为: " . GetBinStr(E_ERROR);
echo "<br />E_WARNING=" . E_WARNING . ", \t\t其对应二进制值为: " . GetBinStr(E_WARNING);
echo "<br />E_NOTICE=" . E_NOTICE . ", \t\t其对应二进制值为: " . GetBinStr(E_NOTICE);
echo "<br />E_USER_NOTICE=" . E_USER_NOTICE . ", \t\t其对应二进制值为: " . GetBinStr(E_USER_NOTICE);
echo "<br />E_ALL=" . E_ALL . ", \t\t其对应二进制值为: " . GetBinStr(E_ALL);
echo "</pre>";
```

运行结果：

E_ERROR=1,	其对应二进制值为: 0000000000000001
E_WARNING=2,	其对应二进制值为: 0000000000000010
E_NOTICE=8,	其对应二进制值为: 0000000000001000
E_USER_NOTICE=1024,	其对应二进制值为: 0000010000000000
E_ALL=30719,	其对应二进制值为: 0111011111111111