jinyPHP 1권 - 11. 전처리기

**11. 전처리기**

간단한 프로그램은 소스 파일 하나에 모든 기능을 구현하는 데 어려움이 없을 것입니다. 하지만 기능이 많아지고 소스가 길어진다면 어떻게 될까요? 소스 파일은 매우 복잡할 것이고 내용을 찾기 위해서는 수많은 스크롤을 해야 할 것입니다.

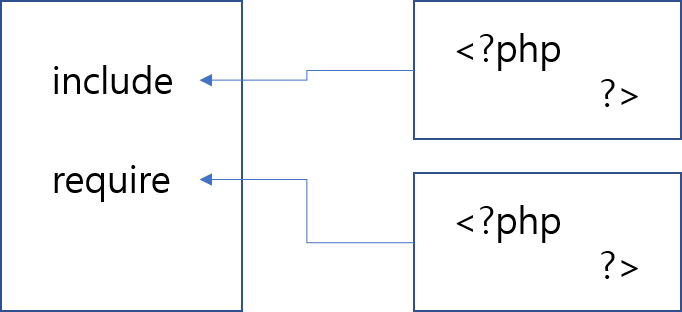
이런 경우 소스의 파일을 기능별로 구분하고, 특별한 전처리기 명령을 통해 소스 파일을 결합할 수 있습니다.

**11.1 include & require**

C 언어와 같은 프로그램은 #include 와 같은 전처리기 기능을 갖고 있습니다. 소스의 필요한 함수나 내용들을 별도의 해더파일에 분리해 놓고 소스 파일들을 결합하여 사용합니다. 이러한 소스의 분할 및 처리는 큰 용량의 프로그램을 개발하는 데 매우 유용합니다.

PHP에도 C 언어의 #include와 같은 전처리 기능을 제공합니다. 하지만 PHP는 이 기능을 두 가지 타입으로 구분하여 제공하고 있습니다. 바로 include와 require 명령입니다.

전처리기 기능을 통해 PHP도 관련 있는 함수와 클래스, 라이브러리 등의 소스들을 별도의 파일로 분리하여 관리할 수 있습니다. 이렇게 분리된 파일들을 include와 require명령어로 다시 소스 파일 결합이 가능합니다.

****

소스 파일들을 분리해 놓으면 여러 사람이 협업하여 개발하기도 편하고 유지보수도 쉽습니다. 또한 코드를 다른 프로젝트에 재사용하기도 편리합니다.

오픈소스 형태의 PHP 라이브러리 또한 이와 같이 별도의 클래스, 함수들로 제작하여 파일 형태로 제공합니다.

PHP는 다음과 같이 네 가지의 전처리 명령을 지원합니다.

* include
* include\_once
* require
* require\_once

**11.2 파일 결합**

include와 require와 같은 전처리기 명령은 PHP 소스 파일들을 읽어와서 서로 결합합니다.

C 언어와 같은 스타일의 프로그램은 소스 실행 상단에 미리 정의를 합니다. C 언어의 해더 파일들은 대부분 함수나 클래스 등으로 구성하기 때문입니다. 하지만 PHP는 전처리기 명령의 위치를 자유롭게 지정하고 파일 결합 위치를 결정할 수 있습니다.

PHP는 C 언어 해더 파일과 달리 함수와 소스 자체를 읽어와서 서로 결합할 수 있기 때문입니다. PHP는 소스 앞에서도 사용할 수 있고 소스 뒤에서도 사용할 수 있습니다. 심지어 소스 중간에서도 include를 통해 파일 삽입, 결합이 가능합니다.

**예제 파일** include-01.php

|  |
| --- |
| **<?php**  **include "header.php";**  **?>**  <html>  <head>  </head>  <body>    **<?php**  **include "index.php";**  **?>**  </body>  </html>  **<?php**  **include "footer.php";**  **?>** |

PHP에서 전처리 명령 include를 통해 소스파일을 삽입하면 전처리기 명령을 선언한 소스 그 자리 안에 지정한 파일의 소스가 삽입됩니다.

어떤 특정한 기능의 파일들은 소스의 삽입 위치가 중요할 수 있습니다. PHP는 소스 자체를 읽어서 삽입을 하기 때문입니다. 하지만 클래스나 함수와 같은 라이브러리 타입의 소스들은 위치에 크게 영향을 받지 않습니다.

대부분 소스파일 분리는 기능적인 부분을 엮어서 파일을 분리합니다(입출력, 파일, 데이터 등).

하지만 워드프레스와 같은 CMS 프로그램은 include 기능을 구조적 레이아웃에 따라 소스를 분리하여 사용하기도 합니다.

**예**

header.php

footer.php

index.php

layout.php

이렇게 구조적으로 분리된 소스 파일들은 유지보수 등을 효율적으로 할 수 있도록 도와줍니다.

**11.3 파일 경로**

전처리기 명령 include 및 require은 삽입하고자 하는 파일의 위치가 매우 중요합니다. PHP의 include와 require명령은 현재 실행되고 있는 PHP 스크립트의 경로를 기준으로 결합하고자 하는 소스파일의 위치를 찾습니다.

만일 실행하고 있는 스크립트와 동일한 폴더 경로에 있는 다른 소스를 사용할 때는 그냥 이름만 넣어도 됩니다. 하지만 다른 폴더나 하위 폴더에 있는 경우에는 상대 경로 또는 절대 경로 형태로 바르게 입력해야 합니다.

/text.php

/lib.php

위처럼 1개의 실행 스크립트 소스파일이 같은 폴더 안에 있습니다. text.php 소스파일은 include(lib.php); 형태로 다른 소스 파일의 경로를 설정하면 됩니다.

/dir1/text.php

/lib.php

위처럼 실행하는 스크립트가 서브 폴더 안에 있습니다. text.php가 실행되면 기본 경로는 /dir1이 됩니다. 따라서 include(../lib.php);처럼 이동한 상위 폴더 경로로 설정해야 합니다.

이처럼 실행 스크립트의 위치에 따라서 경로는 달라집니다. 만일 결합하고자 하는 소스의 파일 경로가 잘못된 경우 오류 메시지를 출력하거나 스크립트 실행이 중단될 수도 있습니다.

**11.3.1 상대 경로**

상대 경로란 현재 디렉터리를 기준으로 파일의 위치를 판별합니다.

자신의 하위 디렉터리에 있는 경우에는 현재 디렉터리(./)를 기준으로 작성하면 됩니다.

./ 파일명 → 현재 디렉터리의 파일명

./폴더1/파일명 → 현재 디렉터리의 하부 폴더1 의 파일

상위 폴더를 선택할 때는 ../를 사용하면 됩니다.

../파일명 → 현재 디렉터리 상위 폴더의 파일명

../../파일명 → 현재 디렉터리의 상위/상위 폴더의 파일명

../상위 폴더명/하부 폴더명/파일명 → 현재 디렉터리의 상위 폴더로 이동한 다음, 상위 폴더명의 하부 폴더명 안에 있는 파일명

**11.3.2 절대 경로**

절대 경로란 컴퓨터 하드디스크의 첫 루트부터의 전체 경로를 의미합니다.

윈도우의 경우,

c:\web\webdoc\test.php

와 같이 드라이브명부터 시작합니다.

리눅스의 경우,

\home\web\webdoc\test.php

와 같이 루트(\)부터 시작합니다.

절대 경로의 장점은 파일의 위치 오류를 방지할 수 있습니다. 하지만, 파일 위치가 변경될 때마다 프로그램 소스상에서 모두 바꿔야 하는 불편함이 있습니다. 또한 절대 경로를 사용하면 경로의 이름이 매우 긴 불편함이 있습니다.

**11.3.3 \_\_DIR\_\_**

PHP 는 현재 자신의 스크립트가 실행되는 경로를 출력해주는 특별한 상수명이 있습니다.

\_\_DIR\_\_ 은 현재 실행되는 스크립트의 경로를 출력합니다. \_\_DIR\_\_ 은 현재 실행되고 있는 스크립트의 서브 디렉터리 경로명까지 포함하여 출력합니다.

**예제 파일** path-01.php

|  |
| --- |
| <?php  echo \_\_DIR\_\_;  ?> |

**결과**

C:\php-7.1.4-Win32-VC14-x86\jinyphp

위의 예에서 \_\_DIR\_\_은 현재 실행하고 있는 스크립트의 유효한 파일 경로를 출력합니다. \_\_DIR\_\_ 상수를 통해 상태 경로를 계산하여 사용하면 유연하게 경로 계산을 할 수 있습니다.

**11.3.4 DocumentRoot**

PHP는 슈퍼 글로벌 서버 변수를 통해 현재 PHP가 실행되는 절대 상위 폴더를 출력합니다.

이전 \_\_DIR\_\_는 실행되는 서브 폴더까지 표시가 되지만, $\_SERVER['DOCUMENT\_ROOT']는 실행 폴더의 root만 표시됩니다.

**예제 파일** path-02.php

|  |
| --- |
| <?php  echo $\_SERVER['DOCUMENT\_ROOT'];  ?> |

**결과**

C:\php-7.1.4-Win32-VC14-x86

위의 예는 root 경로를 출력합니다. PHP의 내부 서버 실행의 문서 루트 경로, 아차피 웹 서버의 문서 루트 경로를 출력합니다.

**11.3.5 \_\_FILE\_\_**

\_\_FILE\_\_ 함수는 현재 실행하고 있는 스크립트의 소스 파일 정보를 출력합니다.

**예제 파일** path-03.php

|  |
| --- |
| <?php  echo \_\_FILE\_\_;  echo "<br>";  // PHP 파일의 절대 서버 경로  echo realpath(\_\_FILE\_\_);  echo "<br>";  // PHP 파일 이름  echo basename(\_\_FILE\_\_);  echo "<br>";  ?> |

**결과**

C:\php-7.1.4-Win32-VC14-x86\jinyphp\path-03.php

C:\php-7.1.4-Win32-VC14-x86\jinyphp\path-03.php

path-03.php

위의 예는 현재 실행하고 있는 스크립트의 소스 파일 정보를 알아낼 수 있습니다. 또한 realpath() 함수를 통해 절대 경로로 변경할 수 있습니다. 만일 경로를 제외한 현재의 실행 스크립트 파일명만 알고 싶다면 basename() 함수를 이용할 수도 있습니다.

**11.3.6 dirname**

dirname() 함수는 입력된 파일 경로에서 디렉터리 부분만 추출할 수 있습니다.

PHP 7.x부터는 입력된 path에서 디렉터리 경로를 추출할 때 경로 레벨을 선택해서 가지고 올 수도 있습니다.

**예제 파일** dirname.php

|  |
| --- |
| <?php  $path = "/dir1/dir2/dir3/3text.txt";    echo dirname($path,1)."<br>";  echo dirname($path,2)."<br>";  echo dirname($path,3)."<br>";  ?> |

**결과**

/dir1/dir2/dir3

/dir1/dir2

/dir1

위의 예는 입력한 $path 경로를 슬래시(/) 기호로 구분하여 각각 접근할 수 있습니다.

**11.4 include**

include 전처리기 명령어는 아주 익숙한 키워드입니다. 다양한 언어에서도 동일한 키워드를 사용하는 경우가 많습니다.

**|문법|**

|  |
| --- |
| include “파일명”'; |

PHP가 스크립트 실행 도중에 include 명령을 만나면 해당 명령이 선언한 자리에 지정한 파일을 삽입, 결합합니다.

**11.4.1 파일 실패**

전처리기 명령어 include는 삽입, 결합할 php 스크립트를 동일한 서버에서 찾습니다. 소스파일을 찾으면 삽입을 통하여 소스를 결합하고, 만일 소스 파일을 찾지 못하면 경고(E\_WARNING)를 출력합니다. 환경설정에서 이 경고는 무시하고 출력하지 않도록 설정할 수도 있습니다.

include명령어는 PHP를 실행하는 데 있어서 파일 존재 여부와 상관없습니다. 만일 include 명령에서 지정한 소스파일을 찾지 못해도 PHP는 정상적으로 실행됩니다. 단지 오류 메시지만 출력합니다.

따라서 정확한 include 명령을 하고 싶다면 파일의 경로나 존재 여부를 개발자가 꼭 확인하고 명령을 작성하는 것이 좋습니다.

예)

|  |
| --- |
| if(file\_exists($filename)){  include $filename;  } |

위 예 표현에서 사용한 file\_exists() 함수는 파일의 존재 유무를 확인합니다. 이는 PHP의 파일 처리 내부 함수입니다.

**11.4.2 중복 처리**

전처리기 include명령어는 삽입하고자 하는 소스 파일의 중복 여부를 체크하지 않습니다.

만일 소스상에서 동일한 include를 중복하여 작성했을 때가 있습니다. 이런 경우 PHP는 파일을 두 번 읽어서 결합을 하게 됩니다.

|  |
| --- |
| include 'sub\_function.php';  $i = $+1;  include 'sub\_function.php'; |

만일 삽입하는 파일 안에 함수와 같은 선언이 있다면 PHP는 바로 오류를 출력할 것입니다.

하지만 인위적으로 단순한 처리 로직을 두 번 처리해야 하는 경우라면 이때도 매우 유용할 수 있습니다.

**예제 파일** include\_lib.php

|  |
| --- |
| <?php  function addPlus($x,$y){  return $x+$y;  }  ?> |

**예제 파일** include-02.php

|  |
| --- |
| <?php  include "include\_lib.php";  $x = 10;  $y = 5;    echo addPlus($x,$y);  ?> |

**결과**

15

위의 예제는 include 명령어를 통해 실행 파일을 2개로 분리했습니다. include-02.php 파일은 include\_lib.php 파일을 삽입하여 실행 함수를 호출합니다. include-02.php에서 사용한 addPlus() 함수는 include\_lib.php 파일에 정의되어 있습니다.

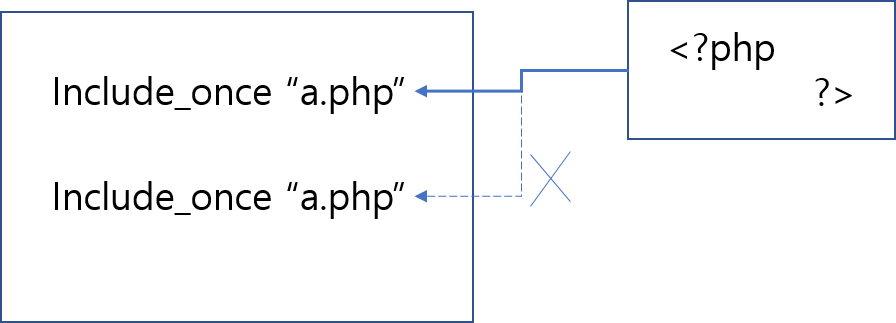
**11.4.3 include\_once**

기존 include명령어는 중복 사용할 경우 여러 문제를 발생할 수 있습니다. 이런 이중 사용으로 인해 발생할 수 있는 문제들을 방지하기 위해서 꼭 한 번만 include해야 되는 스크립트가 있다면 include 대신에 include\_once 사용을 권장합니다.

|문법|

|  |
| --- |
| include\_once 'filename'; |

include\_once명령어는 의미 그대로 소스상에서 지정한 파일을 한 번만 삽입을 하라는 의미입니다.



**예제 파일** include-03.php

|  |
| --- |
| <?php  include\_once "include\_lib.php";  // 두 번 삽입을 해도 오류가 나지 않습니다.  include\_once "include\_lib.php";  $x = 10;  $y = 5;    echo addPlus($x,$y);  ?> |

**결과**

15

위의 예는 include\_one을 이용한 실험입니다. include\_once명령어는 두 번 삽입하는 실수를 방지할 수 있습니다. 중복하여 사용하는 경우 두 번째 include\_once는 무시하게 됩니다.

**11.5 require**

PHP에서는 include와 비슷한 require 전처리 명령어를 지원합니다. require 명령어를 제공하는 이유는 좀 더 유연한 소스 결합과 엄격한 소스 결합을 서로 구분하기 위함입니다.

**11.5.1 Require특징**

PHP는 왜 include와 비슷한 require를 지원하는 것일까요? include와 require의 차이점은 삽입하고자 하는 **파일의 존재 여부에 따른 처리 방법**입니다.

명령 키워드의 의미만으로도 특징을 유추해 볼 수 있습니다. 만일 삽입할 PHP 스크립트 파일이 없는 경우 include 처럼 경고(E\_WARNING)를 출력하는 대신에 require 명령어는 소스의 치명적 오류(E\_COMPILE\_ERROR)를 출력한 후에 스크립트를 즉시 중단합니다.

**|문법|**

|  |
| --- |
| require 'filename'; |

따라서 약간 동적 스타일로 삽입하는 include 대신에 프로그램에서 반드시 삽입 처리가 이루어져야 하는 필수 기능은 require를 사용하는 것을 권장합니다.

**예제 파일** include\_lib.php

|  |
| --- |
| <?php  function addPlus($x,$y){  return $x+$y;  }  ?> |

**예제 파일** require-01.php

|  |
| --- |
| <?php  require "include\_lib.php";  $x = 10;  $y = 5;    echo addPlus($x,$y);  ?> |

**결과**

15

위의 예제는 분리된 2개의 소스를 require 명령으로 결합하는 예입니다. 만일 결합하고자 하는 require 소스 파일이 없으면 실행을 중단하고 오류를 발생할 것입니다.

**11.5.2 require\_once**

require도 중복으로 동일한 파일을 선언 시 두 번 require하는 것을 방지 하기 위해서 include\_once와 유사한 require\_once라는 명령도 같이 제공합니다.

|문법|

|  |
| --- |
| require\_once 'filename'; |

require-01.php 파일은 include\_lib.php 파일을 삽입하여 실행 함수를 호출합니다.

**예제 파일** require-02.php

|  |
| --- |
| <?php  require\_once "include\_lib.php";  // 두 번 삽입을 해도 오류가 나지 않습니다.  require\_once "include\_lib.php";  $x = 10;  $y = 5;    echo addPlus($x,$y);  ?> |

**결과**

15

위의 예는 require\_one를 이용한 예입니다. require\_once명령어는 두 번 삽입하는 실수를 방지할 수 있습니다. 중복하여 사용하는 경우 두 번째 require\_once는 무시하게 됩니다.