




# Inclusión de ficheros externos

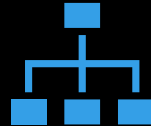




## Inclusión de ficheros externos en PHP



A medida que los programas crecen en tamaño y complejidad, encontrar la información necesaria dentro del código se vuelve más complicado.



Es altamente recomendable organizar el código separando bloques funcionales, como conjuntos de funciones, en ficheros externos. Esto no solo mejora la legibilidad del código, sino que también facilita su mantenimiento y reutilización.



Posteriormente, se puede hacer referencia a estos ficheros externos dentro del programa principal, permitiendo que PHP incluya su contenido como si fuera parte del código actual, utilizando funciones como `include()`, `require()`, `include_once()` o `require_once()`.



## Inclusión de ficheros externos

---

### Funciones

---

---

```
require("ruta/archivo.php");
```

---

```
include("ruta/archivo.php");
```

---


```
require_once("ruta/archivo.php");
```

---

```
include_once("ruta/archivo.php");
```

---

require() vs include()



# **require() vs include()**

Ambas funciones, **require()** e **include()**, permiten insertar el contenido de un archivo PHP dentro de otro.

## ► Diferencias

► **require()** se utiliza cuando el archivo externo es esencial para el correcto funcionamiento del programa. Si el archivo especificado no se encuentra o no puede ser cargado, PHP generará un error fatal (**PHP Fatal error**), **deteniendo** la ejecución del programa de inmediato.

► **include()**, por otro lado, es más flexible. Si no se encuentra el archivo especificado, PHP lanzará solo una advertencia (**Warning**), pero **el programa continuará** ejecutándose.

**require\_once()  
include\_once()**



**vs**

**require()  
include()**

► Las funciones `require_once()` e `include_once()` funcionan de manera similar a sus versiones originales, `require()` e `include()`, respectivamente.

► La diferencia principal es que las versiones con `_once` evitan que un archivo sea cargado más de una vez en el mismo script, lo que previene problemas como la redeclaración de variables, funciones o clases.

► Aunque podría parecer que siempre es mejor usar las versiones con `_once` para evitar errores de duplicación, es importante tener en cuenta que estas funciones son más costosas en términos de rendimiento, ya que PHP necesita verificar si el archivo ya ha sido cargado.

► Por ello, es recomendable utilizarlas únicamente cuando exista la posibilidad de que un archivo se incluya varias veces accidentalmente.



## Inclusión de ficheros externos

- Cuando se incluye un archivo en PHP, el código dentro de dicho archivo hereda el ámbito (scope) de las variables presentes en el punto donde ocurre la inclusión.
  - Esto significa que **cualquier variable disponible en esa línea dentro del archivo que realiza la inclusión también estará disponible para el archivo incluido a partir de ese momento.**
- Si la inclusión del archivo se realiza **dentro de una función**, el código contenido en el archivo incluido también se comportará como si hubiera sido definido dentro de esa función. Por lo tanto, **las variables** del archivo incluirán el ámbito local de la función, y solo serán accesibles dentro de la misma.
- No obstante, todas las **funciones y clases** definidas en el archivo incluido se declaran en el **ámbito global**, independientemente de dónde se realice la inclusión.
  - Esto significa que las funciones y clases estarán disponibles en todo el programa, incluso si el archivo fue incluido dentro de una función.



## variable superglobal

Una variable superglobal en PHP es una variable predefinida que está disponible en todos los contextos del script, sin necesidad de ser pasada como parámetro o declarada como global. Estas variables son accesibles en cualquier parte del código, ya sea dentro de funciones, métodos, o en el ámbito global.

### Características de las Superglobales

**1. Disponibilidad Universal:** Las superglobales están disponibles en cualquier parte del script, sin importar el ámbito en el que te encuentres.

**2. Predefinidas:** Son variables que PHP define automáticamente, por lo que no necesitas inicializarlas.

**3. Acceso Directo:** Puedes acceder a ellas directamente sin necesidad de usar la palabra clave global.





variable  
superglobal

## Principales Superglobales en PHP

- **\$\_GET**: Contiene los datos enviados a través de una solicitud HTTP GET.
- **\$\_POST**: Contiene los datos enviados a través de una solicitud HTTP POST.
- **\$\_REQUEST**: Contiene los datos enviados a través de solicitudes HTTP GET, POST y COOKIE.
- **\$\_SERVER**: Contiene información sobre el servidor y el entorno de ejecución.
- **\$\_FILES**: Contiene información sobre los archivos subidos a través de un formulario.
- **\$\_ENV**: Contiene las variables de entorno.
- **\$\_COOKIE**: Contiene las cookies enviadas por el navegador del cliente.
- **\$\_SESSION**: Contiene los datos de la sesión actual.





## variable superglobal \$\_SERVER

En el caso de `$_SERVER`, esta variable es generada por el servidor web y contiene un array con información relacionada con el entorno de ejecución y la solicitud HTTP actual.

Algunos ejemplos de datos que proporciona incluyen la ruta del script, la dirección IP del cliente, el nombre del servidor y el método de la solicitud.

Importante:

- La información almacenada en `$_SERVER` no puede ser eliminada ni modificada directamente, ya que se trata de datos generados por el servidor en tiempo de ejecución.
- Sin embargo, como cualquier array en PHP, puedes acceder y leer los valores que contiene, y en ciertos casos, puedes agregar o modificar elementos, **pero esto no alterará los datos originales proporcionados por el servidor.**



## variable superglobal \$\_SERVER

Una de las informaciones clave que contiene el array superglobal `$_SERVER` es la **ruta del directorio raíz de documentos** del servidor web, desde donde se está ejecutando el script actual.

Esta información es tomada del archivo de configuración del servidor **Apache**.

```
<?php
echo $_SERVER['DOCUMENT_ROOT'];
//Mostrará la ruta de la carpeta raíz de nuestro servidor
//Por ejemplo: C:/xampp/htdocs
?>
```

### RECOMENDACIÓN

Para garantizar que siempre se incluya el archivo correcto, independientemente de la ubicación del script, puedes utilizar `$_SERVER['DOCUMENT_ROOT']` para definir rutas absolutas.

Es útil cuando el archivo que deseas incluir no está en la misma carpeta que el script:

```
require($_SERVER['DOCUMENT_ROOT'] . '/clase/funciones/archivo.php');
```

En este ejemplo, PHP buscará el archivo en la ruta completa `/clase/funciones/archivo.php` a **partir del directorio raíz del servidor**.



## Rutas relativas a los archivos

### Incluir archivos según su ubicación

- Si el archivo se encuentra en la misma carpeta que el script actual:

```
require('archivo.php');
```

- Si el archivo está en una subcarpeta dentro de la carpeta actual:

```
require('misfunciones/archivo.php');
```

Ten en cuenta que **cuando incluyes archivos en subcarpetas, no debes colocar la barra diagonal al inicio de la ruta**, ya que eso haría que PHP busque la ruta desde el directorio raíz.



## Función dirname()

La función `dirname()` en PHP se utiliza para **obtener la ruta del directorio padre** de una ruta de archivo dada. Es especialmente útil cuando necesitas manipular rutas de archivos y directorios en los scripts.

```
string dirname ( string $path , int $levels = 1 )
```

- `$path`: La ruta del archivo o directorio de la cual deseas obtener el directorio padre.
- `$levels` (opcional): El número de niveles de directorios padres que deseas subir. El valor predeterminado es 1.

```
$archivo = '/var/www/html/index.php';  
$directorioPadre = dirname($archivo);  
echo $directorioPadre; // Salida: /var/www/html  
$directorioPadre = dirname($archivo, 2);  
echo $directorioPadre; // Salida: /var/www
```



## variable superglobal \$\_GET

La superglobal `$_GET` en PHP es un array asociativo que contiene los datos enviados a través de una solicitud HTTP GET. Estos datos son enviados como parte de la URL, después del signo de interrogación `?` Y cada uno de ellos separados por el signo `&`

### Características de `$_GET`

- 1. Acceso a Datos de la URL:** `$_GET` permite acceder a los parámetros de la URL.
- 2. Seguridad:** Los datos enviados a través de GET son visibles en la URL, por lo que no deben usarse para enviar información sensible.
- 3. Limitaciones de Tamaño:** Los datos enviados a través de GET tienen limitaciones de tamaño, dependiendo del navegador y del servidor.



variable  
superglobal  
\$\_GET

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Enlace GET</title>
</head>
<body>
  <h1>Enviar Datos con GET</h1>
  <a href="pruebas.php?nombre=Juan&edad=25">Enviar Datos</a>
</body>
</html>
```

<http://localhost/pruebas.php?nombre=Juan&edad=25>

Archivo php llamado **pruebas.php**

```
<?php
if (isset($_GET['nombre']) && isset($_GET['edad'])) {
  $nombre = $_GET['nombre'];
  $edad = $_GET['edad'];
  echo "Nombre: " . $nombre . "<br>";
  echo "Edad: " . $edad;
} else {
  echo "Parámetros no proporcionados.";
}
```



## variable superglobal \$\_POST

La superglobal `$_POST` en PHP es un array asociativo que contiene los datos enviados a través de una solicitud HTTP POST. Este método es comúnmente utilizado para enviar datos de formularios de manera segura y eficiente.

### Características de `$_POST`

- 1.Seguridad:** Los datos enviados a través de POST no son visibles en la URL, lo que lo hace más seguro para enviar información sensible.
- 2.Sin limitaciones de tamaño:** A diferencia de GET, POST no tiene limitaciones significativas en el tamaño de los datos que se pueden enviar.
- 3.Uso común en formularios:** Es el método preferido para enviar datos de formularios que pueden incluir información sensible o grandes cantidades de datos.





variable  
superglobal  
\$\_POST

```
<!DOCTYPE html>
<html lang="es">

<head>
  <meta charset="UTF-8">
  <title>Formulario GET</title>
</head>

<body>
  <form action="pruebas.php" method="post">
    <label for="nombre">Nombre:</label>
    <input type="text" id="nombre" name="nombre">

    <label for="edad">Edad:</label>
    <input type="number" id="edad" name="edad">

    <input type="submit" value="Enviar">
  </form>
</body>
</html>
```

<http://localhost/pruebas.php?nombre=Juan&edad=25>

Archivo php llamado pruebas.php

```
<?php
if (isset($_POST['nombre']) && isset($_POST['edad'])) {
  $nombre = $_POST['nombre'];
  $edad = $_POST['edad'];
  echo "Nombre: " . $nombre . "<br>";
  echo "Edad: " . $edad;
} else {
  echo "Parámetros no proporcionados.";
}
```