



# PHP – POO Namespaces

---



**Ciclo:** DAW  
**Módulo:** DWES  
**Curso:** 2020-2021  
**Autor:** César Guijarro Rosaleny



**Introducción.....3**

**Definir espacios de nombres.....4**

**Uso de los espacios de nombre.....5**

**Importar espacios de nombre.....10**

**Bibliografía.....13**

## Introducción

---

Los **espacios de nombres (namespaces)** son una manera de encapsular elementos. Se pueden ver como un concepto abstracto en muchos aspectos.

Por ejemplo, en cualquier sistema operativo, los directorios sirven para agrupar ficheros relacionados, actuando así como espacios de nombres para los ficheros que contienen. Como ejemplo, el fichero `foo.txt` puede existir en los directorios `/home/greg` y `/home/otro`, pero no pueden coexistir dos copias de `foo.txt` en el mismo directorio. Además, para acceder al fichero `foo.txt` fuera del directorio `/home/greg`, se debe anteponer el nombre del directorio al nombre del fichero, empleando el separador de directorios (`/`) para así obtener `/home/greg/foo.txt`. Este mismo principio se extiende a los espacios de nombres en el mundo de la programación.

En el mundo de PHP, los espacios de nombres están diseñados para solucionar dos problemas con los que se encuentran los autores de bibliotecas y de aplicaciones al crear elementos de código reusable, tales como clases o funciones:

- El conflicto de nombres entre el código que se crea y las clases/funciones/constantes internas de PHP o las clases/funciones/constantes de terceros.
- La capacidad de apodar (o abreviar) Nombres\_Extra\_Largos diseñada para aliviar el primer problema, mejorando la legibilidad del código fuente.

## Definir espacios de nombres

---

Los espacios de nombres se declaran utilizando la palabra reservada **namespace**. Un fichero que contenga un espacio de nombres debe declararlo **al inicio del mismo, antes que cualquier otro código**, con una excepción: la palabra reservada **declare**.

```
<?php  
  
namespace miProyecto;  
  
function hola() {  
    echo "Hola mundo";  
}
```

Además, **todo lo que no sea código de PHP no puede preceder a la declaración del espacio de nombres, incluyendo espacios en blanco extra**.

También, a diferencia de otras construcciones de PHP, **se puede definir el mismo espacio de nombres en varios ficheros**, permitiendo la separación del contenido de un espacio de nombres a través del sistema de ficheros

Al igual que los directorios y ficheros, los espacios de nombres de PHP también tienen la capacidad de especificar una jerarquía de nombres de espacios de nombres. Así, un nombre de un espacio de nombres se puede definir con subniveles:

```
<?php  
  
namespace miProyecto\Sub\Nivel;
```

## Uso de los espacios de nombre

---

Se puede hacer una simple analogía entre los espacios de nombres de PHP y el sistema de ficheros. Existen tres maneras de acceder a un fichero en el sistema de ficheros:

- **Nombre de fichero relativo como *foo.txt*.** Se resuelve con *directorio\_actual/foo.txt* donde *directorio\_actual* es el directorio actualmente ocupado. Así, si el directorio actual es */home/foo*, el nombre se resuelve con */home/foo/foo.txt*.
- **Nombre de ruta relativa como *subdirectorio/foo.txt*.** Se resuelve con *directorio\_actual/subdirectorio/foo.txt*.
- **Nombre de ruta absoluta como */main/foo.txt*.** Se resuelve con */main/foo.txt*.

Se puede aplicar el mismo principio a los elementos del espacio de nombres de PHP.

Vamos a ver un ejemplo para entender como funciona. Crea dos carpetas en *apache\_htdocs/DWS/ejemplos/T4/namespaces* llamadas *carpeta1* y *carpeta2*. Dentro de *carpeta1* tendremos dos archivos (*archivo1.php* y *archivo2.php*) y una carpeta llamada *subcarpeta1*. Ésta última carpeta (*subcarpeta1*) tendrá otro archivo llamado *archivo3.php*. Por último, crea el archivo *archivo4.php* dentro de la carpeta *carpeta2*.

En principio, todos los archivos tendrán el mismo contenido: el namespace al que pertenece y una función llamada *hola()*.

Para simplificar, El namespace será el nombre de la carpeta donde está ubicado el archivo: *carpeta1* para los archivos 1 y 2, *carpeta1\subcarpeta1* para el archivo 3 y *carpeta2* para el archivo 4.

La función *hola* mostrará la frase “*Hola mundo del nombre\_archivo*”. Por ejemplo, el código de *archivo3.php* será:

```
<?php

namespace carpeta1\subcarpeta1;

function hola() {
    echo "Hola mundo de archivo3<br>";
}
```

Ahora vamos a modificar *archivo1.php* para ir añadiendo el resto de archivos y usar los namespaces para ejecutar la función *hola()* de los diferentes ficheros. Primero incluye *archivo2.php*:

```
<?php

namespace carpeta1;

include "archivo2.php";

function hola() {
    echo "Hola mundo de archivo3<br>";
}
```

Al abrir el archivo en un navegador nos mostrará un error, ya que los dos archivos pertenecen al mismo namespace y tienen dos funciones que se llaman igual, lo cual no está permitido.

Fatal error: Cannot redeclare carpeta1\hola() (previously declared in /var/www/html/DWS/ejemplos/T4/namespaces/carpeta1/archivo1.php:9) in /var/www/html/DWS/ejemplos/T4/namespaces/carpeta1/archivo2.php on line 5				
Call Stack				
#	Time	Memory	Function	Location
1	0.2055	398320	{main}()	.../archivo1.php:0

Cambia el nombre de la función de *archivo2.php* por *hola2()* y ejecuta ambas funciones:

```
<?php

namespace carpeta1;

include "archivo2.php";

function hola() {
    echo "Hola mundo de archivo3<br>";
}

hola();
hola2();
```

Ahora debería mostrarte las dos frases en el navegador:



A la hora de ejecutar la función, PHP lo traducirá por *carpeta1\hola()* y *carpeta1\hola2()*, ya que *archivo1.php* pertenece al namespace *carpeta1*.

El siguiente paso será incluir *archivo3* y ejecutar su función:

```
<?php

namespace carpeta1;

include "archivo2.php";
include "subcarpeta1/archivo3.php";

function hola() {
    echo "Hola mundo de archivo3<br>";
}

hola();
hola2();
subcarpeta1\hola();
```

Igual que en los sistemas de ficheros, PHP entenderá que lo que ejecutamos en la última línea pertenece al namespace *carpeta1\subcarpeta1* (es similar a utilizar rutas relativas en los sistemas de ficheros).

Por último, vamos a añadir *archivo4* y ejecutar su función. En este caso tendremos que indicar el namespace completo, ya que *carpeta2* no es un subnivel del namespace *carpeta1* (sería como utilizar rutas absolutas en un sistema de ficheros).

Para hacerlo, debemos empezar el namespace de la función a ejecutar con una barra invertida.



```
<?php

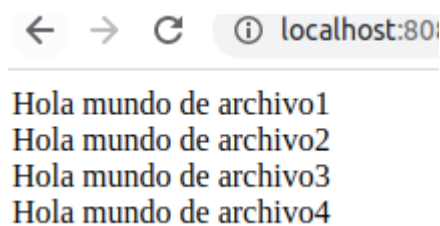
namespace carpeta1;

include "archivo2.php";
include "subcarpeta1/archivo3.php";
include "../carpeta2/archivo4.php";

function hola() {
    echo "Hola mundo de archivo3<br>";
}

hola();
hola2();
subcarpeta1\hola();
\carpeta2\hola();
```

Si ejecutamos el archivo, vemos que nos muestra la salida de las 4 funciones.



```
← → ↻ ⓘ localhost:8080

Hola mundo de archivo1
Hola mundo de archivo2
Hola mundo de archivo3
Hola mundo de archivo4
```

## Importar espacios de nombre

---

Uno de los usos más habituales de los namespaces es junto a la POO, para utilizar clases definidas en otros archivos. El problema es que tienes que anteponer el namespace al que pertenece esa clase siempre que quieras utilizarla.

Por ejemplo, crea dos carpetas (*carpeta1* y *carpeta2*) en *apache\_htdocs/DWS/ejemplos/T4/namespaces2*. Dentro de *carpeta1* crea el archivo *archivo1.php* y dentro de *carpeta2* el archivo *archivo2.php*.

En *archivo2.php* vamos a definir una clase cualquiera, por ejemplo:

```
<?php

namespace carpeta2;

class miClase {

function __construct() {
    echo "objeto de la clase miClase creado";
}

}
```

Si queremos utilizar esa clase en *archivo1.php* tenemos que incluir el archivo donde está definida y usar *carpeta2\miClase* cada vez que la referenciamos.

```
<?php

namespace carpeta1;

include ('../carpeta2/archivo2.php');

$objeto1 = new \carpeta2\miClase();
```

Podemos simplificar el proceso importando la clase con su espacio de nombre usando la palabra reservada **use**. De esta forma, ya no hará falta anteponer el namespace al que pertenece esa clase:

```
<?php

namespace carpeta1;
use carpeta2\miClase;

include ('../carpeta2/archivo2.php');

$objeto1 = new miClase();
```

También podemos utilizar un alias para la clase importada usando la palabra reservada **as**.

```
<?php

namespace carpeta1;
use carpeta2\miClase as claseNueva;

include ('../carpeta2/archivo2.php');

$objeto1 = new claseNueva();
```

Aunque pueda parecer que los espacios de nombres no tienen mucha utilidad, veremos como la mayoría de frameworks los utilizan combinados con el autoloader para importar las clases sin necesidad de incluir ningún archivo.

## Bibliografía

---

<https://www.php.net/manual/es/index.php>