Índex

[**¿Cómo hacer debugging en PHP?** 1](#_Toc108380324)

[**La función dd() en PHP** 1](#_Toc108380325)

[**¿Qué son las cookies?** 1](#_Toc108380326)

[**¿Qué son las sesiones?** 2](#_Toc108380327)

[**¿Qué son las excepciones?** 3](#_Toc108380328)

[**¿Qué es un error en tiempo de ejecución?** 3](#_Toc108380329)

[**¿Para qué sirven las excepciones?** 3](#_Toc108380330)

[**Notros también podemos lanzar errores** 3](#_Toc108380331)

[**La interfaz Throwable** 4](#_Toc108380332)

[**¿Cómo trabajar con fechas en PHP?** 4](#_Toc108380333)

[**Clase Datetime** 4](#_Toc108380334)

[**Modularización del código** 4](#_Toc108380335)

[**¿Para qué nos sirve?** 4](#_Toc108380336)

[**Include, include\_once, require, require\_once** 5](#_Toc108380337)

[**Namespaces** 5](#_Toc108380338)

[**¿Quiénes son afectados por los espacios de nombres?** 6](#_Toc108380339)

[**PSR-4 y Composer** 6](#_Toc108380340)

[**¿Qué es PSR?** 6](#_Toc108380341)

[**¿Qué es composer?** 7](#_Toc108380342)

[**Front Controller** 7](#_Toc108380343)

[**El archivo .htaccess** 7](#_Toc108380344)

[**Traits** 9](#_Toc108380345)

# **¿Cómo hacer debugging en PHP?**

Existen ciertas funciones predefinidas por PHP como por ejemplo **var\_dump** o **print\_r**.

Pero en frameworks como Laravel podemos llegar a tener herramientas más avanzadas de debugging como la función **dd()**.

# **La función dd() en PHP**

Es una fusión entre la función **var\_dump** y **die**. La función **dd**() no se encuentra integrada nativamente a PHP, sino que forma parte del Core de **Laravel**.

Afortunadamente, podemos replicarlas en PHP con el comando **composer require symfony/var-dumper**

# **¿Qué son las cookies?**

Las cookies son pedazos de memoria de tu computadora que le prestas a cualquier página web para que ellos puedan guardar información personalizada acerca de ti.

La variable superglobal **$\_COOKIE** para obtener las cookies guardadas.

Métodos **setcookie()** o **setrawcookie()** para definir una cookie.

Parámetros que admite el método setcookie():

* **name**: Define el nombre de la cookie.
* **value**: Define el valor de la cookie.
* **expires\_or\_options**: El tiempo en que la cookie expira.
* **path**: La ruta dentro del servidor en la que la cookie estará disponible.
* **domain**: El (sub)dominio al que la cookie está disponible.
* **secure**: Indica que la cookie sólo debiera transmitirse por una conexión segura HTTPS desde el cliente.
* **httponly**: Cuando es **true** la cookie será accesible sólo a través del protocolo HTTP, es decir, lenguajes como JavaScript puede ser accedida. Sí es **False** sólo puede ser accedida desde el server.

# **¿Qué son las sesiones?**

Son muy similares a las **cookies**, pero estas **nos permiten implementar sistemas de autenticación** dentro de un sitio web. Una sesión es una cookie temporal y encriptada que estará viva y contendrá toda la información del usuario mientras este activa.

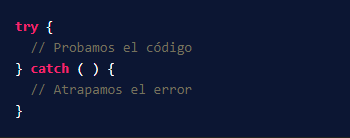
**Las sesiones se destruyen en cuanto hacemos logout**. Con las sesiones podemos tener información especifica de un usuario para personalizar el contenido que le mandamos.

* La función **session\_start()** nos permite comenzar a usar sesiones.
* Tenemos la variable superglobal **$\_SESSION** que es un arreglo. Es único por usuario.
* Para cerrar una sesión usamos el método **session\_destroy()**

Debemos tener cuidado de no usar dos veces la función **session\_start()** dentro de un mismo scripts php ya que esto puede provocar un error.

# **¿Qué son las excepciones?**

Mediante la estructura **try**/**catch** podemos controlar los errores que surjan en **tiempo de ejecución**. Esta es una estructura que nos permite **atrapar** errores, es decir, cada vez que PHP lanza una excepción, esta estructura los atrapa.



## **¿Qué es un error en tiempo de ejecución?**

**Es un error que se lanza cuando el programa está corriendo**, es decir, de inicio no nos da ningún error, el lenguaje o editor de código no marca ningún error, pero a medida que el programa va corriendo surja un error. Por ejemplo:

* Hacer una división entre 0.
* Acceder a un elemento de un array vacío.
* Consultar una URL incorrecta.

## **¿Para qué sirven las excepciones?**

Como programadores nunca debemos dejar que nuestros usuarios vean nuestros errores. Estas excepciones nos permiten mantener los errores bajo control, es decir, nosotros podemos decidir qué hacer con ese error.

## **Notros también podemos lanzar errores**

PHP nos permite lanzar errores. Podemos lanzar errores de forma voluntaria para después atraparlos.

Esto es útil sobre todo cuando creamos errores personalizados.

Únicamente necesitamos usar la palabra **throw**.

## **La interfaz Throwable**

Todas las excepciones dentro de PHP deben implementar la interfaz **Throwable** para poder funcionar. La clase Exception de PHP la implementa directamente.



# **¿Cómo trabajar con fechas en PHP?**

Casi toda la magia la hace la función date. A esta función simplemente hay que pasarle un string con lo que queremos y mágicamente no da ese dato.

## **Clase Datetime**

Cn esta clase podemos obtener varios métodos que nos permitirán trabajar con fechas.

# **Modularización del código**

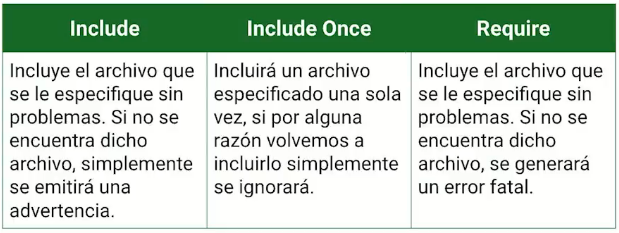
## **¿Para qué nos sirve?**

Cuando trabajamos profesionalmente solemos generar archivos muy **grandes**. Muchas veces esto se vuelve difícil de leer, modularizar el código ayuda a facilitar la lectura.

Esto nos ayuda a mantener el orden y limpieza de nuestro código.

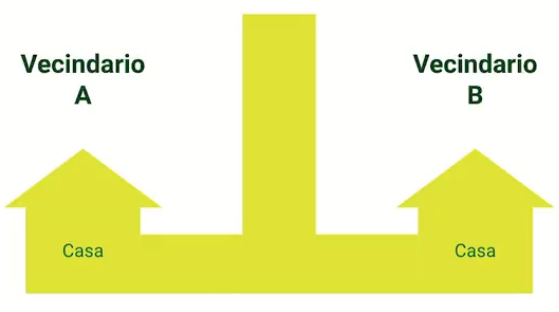
## **Include, include\_once, require, require\_once**

Estas funciones nos permiten **separar** nuestros archivos de código y a requerirlos según necesitemos. Todas tienen el mismo objetivo con ligeras diferencias entre ellas.



# **Namespaces**

También conocidos como **espacios de nombre**, son los **apellidos** que les podemos dar a una **clase**, es decir, podemos tener dos clases con el mismo nombre, pero distintos apellidos (espacios de nombres).



## **¿Quiénes son afectados por los espacios de nombres?**

Aunque podemos escribir cualquier código dentro de un espacio de nombre, los únicos tipos de códigos que serán afectados son las **clases** (incluyendo las abstractas y traits), **interfaces**, **funciones** y **constantes**.

# **PSR-4 y Composer**

## **¿Qué es PSR?**

Son un conjunto de reglas o estándares que definen como trabajar con el lenguaje.

**PSR-4** hace referencia a que, todos los espacios de nombres deben hacer referencia hacia la carpeta en donde dichos archivos están ubicados.

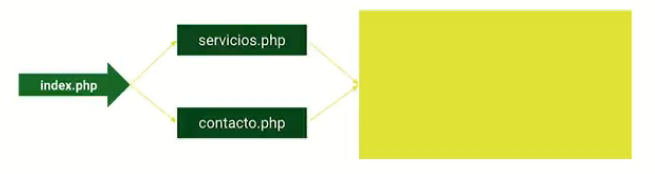
**Composer** usa este PSR para lograr la autocarga de sus archivos, es decir, ya no tenemos que preocuparnos de usar include y require, composer lo hace por nosotros gracias a **PSR-4**.

## **¿Qué es composer?**

**Composer** es el administrador de paquetes de PHP, así como es **npm** en nodejs o **apt** en linux.

# **Front Controller**

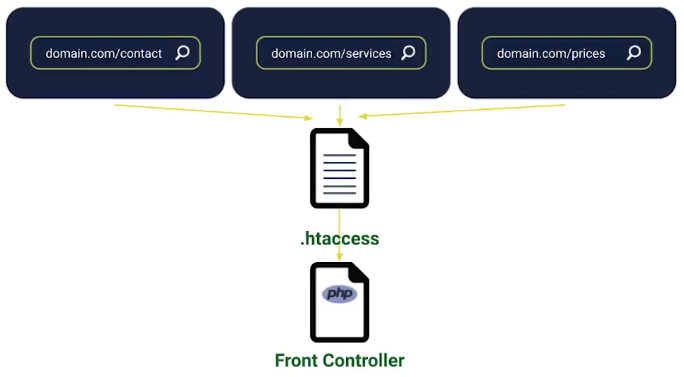
Un Front Controller actúa como un **único punto de entrada** a una página web desde la cual se pueden cargar las páginas solicitadas.



Sí vemos la imagen anterior, **index.php** siempre será nuestro punto de entrada y desde allí podemos cargar distintos archivos, no importa cuales sean.

# **El archivo .htaccess**

Gracias a este archivo podemos tener URLs más limpias, fáciles de leer y buenas para SEO, es decir, son URLs optimizadas para **S**earch **E**ngine **O**ptimization (Optimización para motores de búsqueda).



Sí vemos la imagen anterior, tenemos tres URLs, todas ellas van a pasar por el archivo .**htacces** y este le va a pasar toda la información al **Font-controller**.

En simple palabras, lo que hará el .**htacces** es traducir las URLs que están muy limpias y legibles a algo que el **Front-controller** lo pueda leer.

Para poder trabajar con este archivo debemos tener habilitado la sobrescritura de URLs. Para eso debemos buscar en nuestro SO donde se encuentra ubicado el archivo **httpd.conf** o **apache2.conf**, esto dependerá del SO que se tenga instalado.

Mas información [aquí](https://www.bonaval.com/kb/sistemas-operativos/linux-sistemas-operativos/ejemplos-de-htaccess-que-todo-webmaster-deberia-conocer) y [acá](https://helponnet.com/2019/05/15/mod_rewrite_flags/).

# **Traits**

Desde su versión 5.4.0, PHP implementa una metodología de reutilización de código llamada **Traits**. Un **Traits** es una **plantilla de métodos que podemos usar en diferentes clases**. Es decir, podemos crear un **conjunto de métodos** que podemos usar en varias clases a la vez.

Los **Traits** es una alternativa a una herencia tradicional.