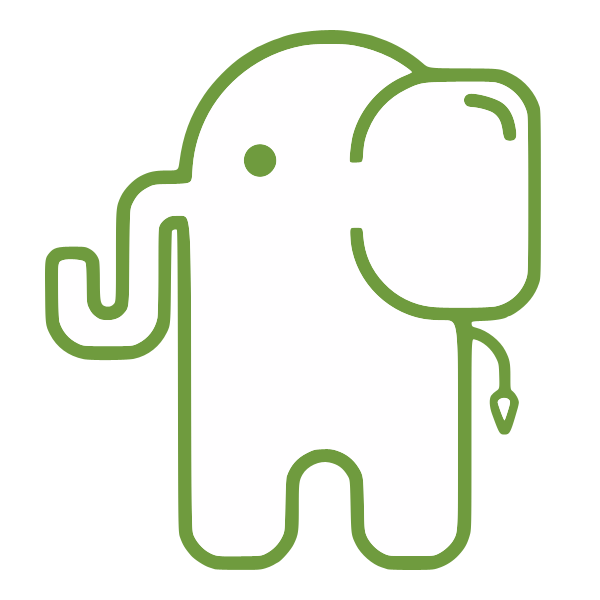
****

**SLIM**

**Framework de PHP**

**Apuntes y Documentación**

**1 COMANDOS PRINCIPALES**

**composer require slim/slim 🡪** Descarga la última versión de Slim para el proyecto.

**composer require slim/slim:2.3 🡪** Descarga una versión específica de Slim para el proyecto.

La última versión de Slim (la v4) necesita de un paquete adicional para darle un nombre a las clases y establecer rutas entre ellas a nivel funcional, llamado **psr7**.

La v3 utilizaba **psr4**.

**composer require slim/psr7 🡪** Descarga lo necesario para utilizar **psr7**.

**composer require slim/twig-view 🡪** Descarga el motor de plantillas de twig (Esto solamente si se va a trabajar con HTML, si se va a utilizar otro motor como *“Vue”*, no hace falta),.

**// Todo lo anteriormente mencionado se guarda en la carpeta “vendor”.**

**php -S localhost:[puerto] 🡪** Arranca el proyecto asignado a dicho puerto.

**2 NECESIDADES BÁSICAS**

# ***1.1- Index.php y referencia al autoload.php***

En el proyecto, vamos a tener un archivo que actúe como *“main”,* que en este caso va a llamarse **index.php** .

Una vez dentro de index.php, lo primero que debemos hacer es referenciar al archivo **autoload.php** de la carpeta *vendor*, ya que este se encarga de cargar todo lo descargado previamente:

require(\_\_DIR\_\_ ."/vendor/autoload.php");

(La constante mágica *“\_\_DIR\_\_”* hace referencia al directorio actual).

# ***1.1- Configurar la instancia de Slim***

Seguido, tenemos que crear una variable que maneje todo lo guardado en el AppFactory de Slim. En nuestro caso la llamaremos **$app**:

$app = AppFactory::create();

# ***1.1- Arrancar el proyecto***

Simplemente integramos en nuestro código la línea:

$app->run();

y ahora el comando del primer apartado para iniciar el proyecto (**php -S localhost:[puerto]**)debería funcionar.

**3 ENDPOINTS**

En SLIM, todos los endpoints necesitan aparte de la ruta URL que queramos asignarles al ejecutar el programa, otros 3 argumentos:

* **Request:** El objeto **Request** contiene toda la información sobre la solicitud HTTP que el cliente ha hecho. Por ejemplo, los datos enviados por el cliente, los encabezados, la URL o los parámetros de consulta.
* **Response:** El objeto **Response** se utiliza para configurar y devolver la respuesta que será enviada al cliente **LOS ENDPOINTS EN PHP SIEMPRE DEVUELVEN UN RESPONSE EN EL return.**
* **array:** Es un array que contiene cualquier parámetro de la ruta que pudiera ser dinámico (como: */user/{id}*).

Los 2 primeros objetos son en realidad: **ServerRequestInterface** y **ResponseInterface**, que son interfaces definidas en *Psr7* que estandarizan cómo manejar las respuestas y las solicitudes HTTP, a las que llamamos *Request* y *Response* respectivamente.

Todo quedaría de la siguiente manera:

<?php

use Psr\Http\Message\ResponseInterface as Response;

use Psr\Http\Message\ServerRequestInterface as Request;

use Slim\Factory\AppFactory;

require(\_\_DIR\_\_ ."/vendor/autoload.php");

$app = AppFactory::create();

$app->get("/", function (Request $request, Response $response, array $args) {

$response->getBody()->write("Página Principal");

return $response;

});

$app->run();

# ***1.1- Sintaxis y aclaraciones***

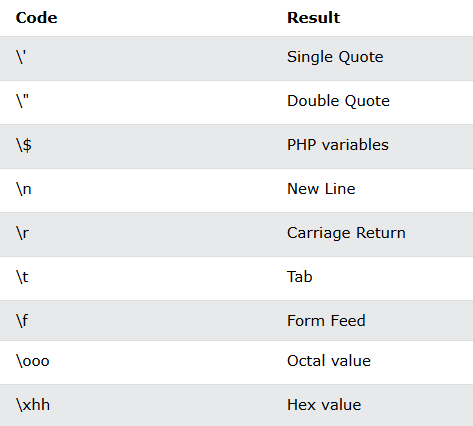
* La extensión de un documento PHP es **.php**
* Se puede combinar en un documento .php tanto HTML como PHP a la vez.
* Los comentarios en una sola línea se escriben con **//** o #.
* // Comentario de una línea. # Comentario de una línea.
* /\* Comentario multilínea. \*/
* Un script de PHP empieza con **<?php** y acaba con **?>**.

<?php

// PHP code goes here.

?>

**Caracteres especiales o caracteres de escape (Escape Characters):**

****

# ***1.2- Sensibilidad***

En PHP, palabras clave (por ejemplo: if, else, while, o echo), clases, funciones y las funciones definidas por el usuario **NO SON SENSIBLES**.

En el siguiente ejemplo, *ECHO* es lo mismo que *echo* y que *EcHo*: