

Primera sesión: programación en PHP

Diseño de Sistemas Software



0. Preparación

Crea una carpeta llamada `sesion01` en tu carpeta de usuario. **Recuerda que en los laboratorios se borra todo el contenido que hayas creado al cerrar la sesión o apagar el ordenador.** Guárdate una copia de esta carpeta antes de terminar la sesión si quieres conservar tu trabajo.

Inicia Visual Studio Code y abre la carpeta que acabas de crear. Se mostrará la carpeta actual y su contenido en el panel de la izquierda, aunque ahora mismo estará vacía.

Nota: puedes abrir una carpeta en VS Code desde el terminal escribiendo `code [ruta]`

```
$ cd sesion01
$ code .
```

A continuación entra en la vista de depuración pulsando el icono  de VS Code. Pulsa el botón `Abrir launch.json` ( en la parte superior de la pantalla) y selecciona el entorno PHP. Esto creará un archivo `.vscode/launch.json` en la carpeta del proyecto con una configuración por defecto para depurar aplicaciones PHP. Cierra este archivo sin cambiar nada.

Ahora tendrás dos opciones disponibles en el desplegable a la izquierda del botón:

- `Listen for XDebug`: queda a la espera de que el programa se ejecute desde otro sitio (p.ej. el terminal). Al estar XDebug activado, si se ejecuta alguno de los archivos que tenemos abiertos, la ejecución se detendrá al llegar a un punto de interrupción y podremos depurar el programa usando VS Code. **Éste es el modo de funcionamiento que usaremos más adelante en los proyectos.**
- `Launch currently open script`: ejecuta el archivo actual usando el intérprete de PHP en línea de comandos con el modo de depuración activado. Es un atajo para no tener que lanzar manualmente los scripts desde el terminal, pero no es apto para probar aplicaciones complejas.

Se pueden crear configuraciones personalizadas en el archivo `launch.json`, por ejemplo para lanzar el script de inicio de una aplicación independientemente del archivo que esté abierto. Para saber cómo hacerlo puedes consultar la documentación de VS Code.

1. ¡Hola mundo!

Crea un archivo llamado `holamundo.php` y copia el siguiente código PHP:

```
<?php
    $saludo = "¡Hola mundo!\n";
    echo $saludo;
?>
```

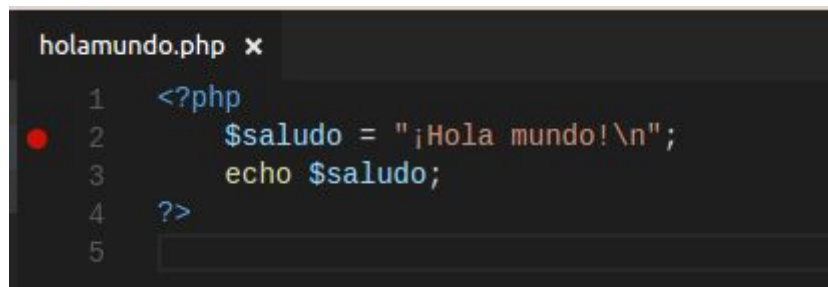
(Lleva cuidado si copias y pegas, es posible que se estén usando comillas tipográficas)


Abre un terminal y cambia la carpeta de trabajo a la carpeta `sesion01`. Ejecuta el archivo que acabas de crear con la siguiente instrucción:

```
$ php holamundo.php
```

2. Depuración con VS Code

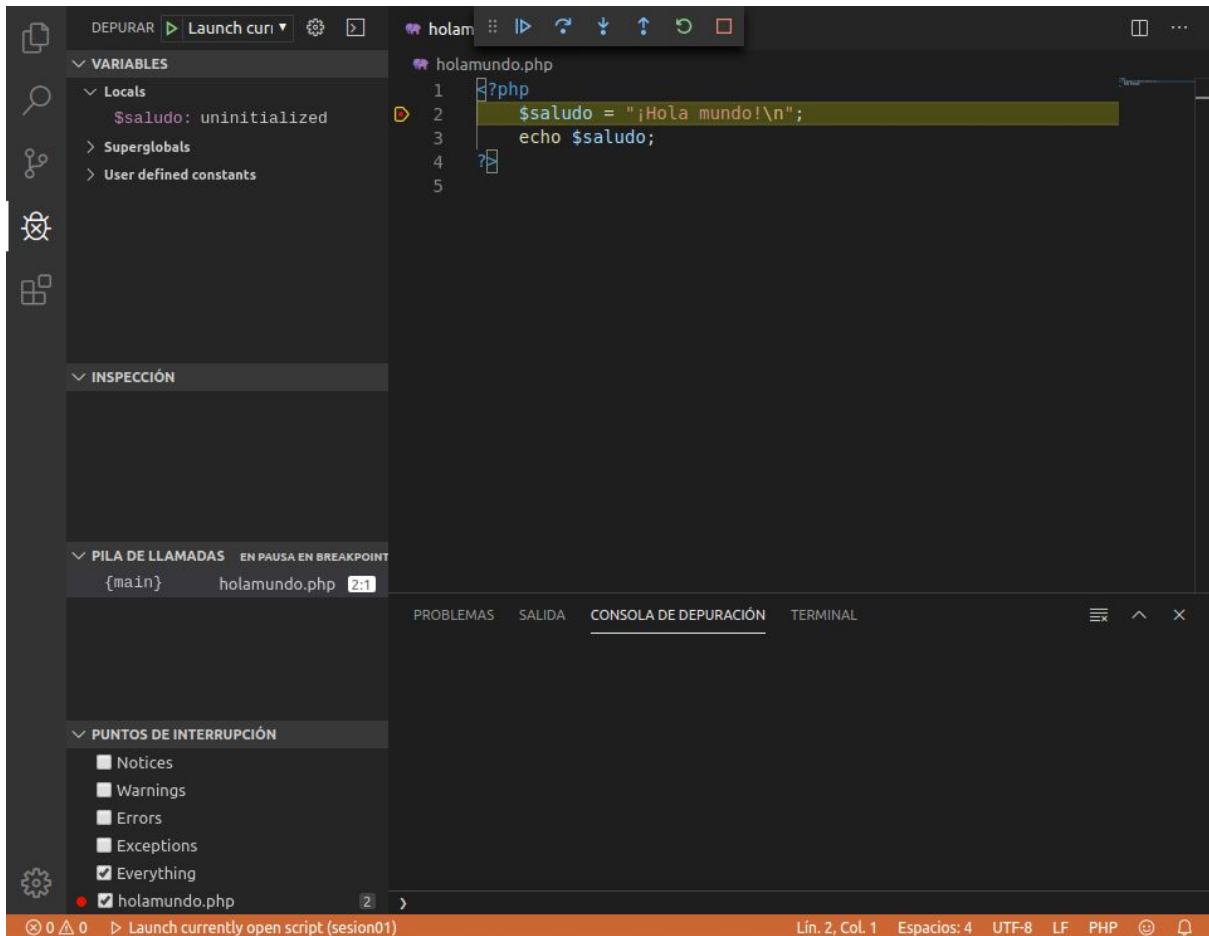
Establece un punto de interrupción en la segunda línea, haciendo click en el área que hay a la izquierda del número de línea en el editor de VS Code.



Ahora ve a la vista de depuración, selecciona el modo `Launch currently open script` y ejecuta el script pulsando el botón `Iniciar depuración` (). La ejecución se detendrá en el punto de interrupción, justo antes de ejecutar esa línea. Observa las herramientas disponibles:

- **Panel de herramientas para controlar la ejecución** (arriba): permite reanudar, parar o ejecutar paso a paso.
- **Variables**: muestra el valor de todas las variables en el contexto actual.
- **Inspección**: permite añadir variables concretas para ver su valor en todo momento, útil cuando hay muchas variables en el contexto actual. También permite añadir llamadas a funciones y expresiones complejas (p.ej. `count($arr)` o `$i+1`).
- **Pila de llamadas**: muestra la pila de funciones para el contexto actual.

- **Puntos de interrupción:** permite activar o desactivar para qué archivos o tipos de eventos se debe detener la ejecución.
- **Consola de depuración:** muestra la salida del programa y permite evaluar fragmentos de código escribiéndolos en la línea inferior.



3. Ejercicios

3.1. Operación división

Escribe una función PHP llamada `divide` que reciba como entrada dos números y devuelva como salida el resultado de su división. En caso de que el divisor sea cero la función tendrá que devolver el valor `null`.

3.2. Cadenas

Escribe una función PHP llamada `invertirCadena` que reciba como parámetro de entrada una cadena y devuelva como salida la misma cadena pero en orden inverso.

Nota: hay una función en PHP que hace esto. Búscala, pero intenta implementar la función manualmente si no estás familiarizado con el lenguaje.

3.3. Ordenación de un array

Escribe una función `ordenacionBurbuja` en PHP que implemente el algoritmo de ordenación por el método de la burbuja a partir del siguiente pseudocódigo. La función debe devolver una copia del array con los elementos ordenados.

```
func bubblesort( var a as array )
    for i from 1 to N
        for j from 0 to N - 1
            if a[j] > a[j + 1]
                swap( a[j], a[j + 1] )
    end func
```

La función auxiliar `swap` debe realizar el intercambio pasándole los dos elementos del array por referencia.

3.4. Función de Fibonacci

Escribe una función PHP llamada `fibonacci` que implemente la sucesión de Fibonacci. Esta sucesión se define como:

$$f_0 = 0; \quad f_1 = 1; \quad f_n = f_{n-1} + f_{n-2} \text{ para } n > 1$$

La función recibirá como parámetro de entrada un número entero `n` y tendrá que devolver como salida el `n`-ésimo valor de la sucesión, o `null` si `n < 0`.

3.5. Fechas

Escribe una función `proximoMes` que devuelva una cadena en formato día/mes/año con la fecha resultante de sumarle un mes a la fecha actual.

Nota: puedes consultar la documentación de la clase `DateTime` aquí que contiene operaciones útiles para realizar esta operación:

<http://php.net/manual/es/class.datetime.php>

3.6. Clases

Escribe una clase llamada `Fibonacci` que guarde en un atributo (array) privado la secuencia de Fibonacci más larga que se le haya pedido. Al crear la clase, en el constructor se calcularán los 10 primeros números de la secuencia ($n=9$) y se almacenarán en el array. Esta clase tendrá además dos métodos públicos:

- `imprimirSecuencia`: imprimirá la secuencia almacenada en el array, con los valores separados por comas, p.ej. `0,1,1,2,3,5,8,13,21,34`

Pista: existe una función en PHP que permite crear una cadena a partir de un array.

- `fibonacci`: igual que en el ejercicio 3.4, pero antes de calcular un valor debe comprobar si ese valor ya ha sido calculado previamente. De ser así devolverá el valor almacenado sin realizar ningún cálculo. En caso contrario calculará y almacenará los valores de la serie faltantes hasta llegar al solicitado.

3.7. Espacios de nombres

Guarda el código del ejercicio anterior en un archivo con el espacio de nombres `dss\ejercicios`. Crea otro archivo que lo incluya y use la clase `Fibonacci`.