Desarrollo web en entorno servidor

Versión 1.0

Ricardo Pérez López

noviembre de 2017

Índice general

Parte I

Introducción

CAPÍTULO 1

Preparación del entorno de desarrollo

1.1 Instalación automatizada

1.1.1 Acciones previas

Instalar `git`

Crear cuenta en GitHub

Usar https://github.com/ricpelo/conf y seguir las instrucciones del README.md

1.2 Terminal

1.2.1 Zsh

1.2.2 Oh My Zsh

1.3 Navegador

1.3.1 Herramientas de desarrollador

1.4 Editores de texto

1.4.1 Vim y less

1.4.2 Atom

CAPÍTULO 2

Introducción al desarrollo web

2.1 Conceptos básicos

2.1.1 Navegadores y servidores web

2.1.2 Agentes de usuario

2.1.3 Web estática vs. dinámica

2.1.4 Estructura vs. contenido

2.2 Ejemplos de aplicaciones web

2.2.1 Redes sociales: Facebook, Twitter...

2.2.2 Comercio electrónico: Amazon, eBay...

2.2.3 Administración electrónica...

2.2.4 Portales

2.2.5 ERP, CRM

2.3 Tecnologías de desarrollo de aplicaciones web

2.3.1 .NET

2.3.2 Java

2.3.3 Ruby/Rails

2.3.4 Python/Django

2.3.5 PHP

Odoo

PrestaShop

Drupal

WordPress

CAPÍTULO 3

Protocolo HTTP y lenguaje HTML

3.1 Arquitectura cliente/servidor

3.2 HTML 5 básico

3.3 Protocolo HTTP

3.3.1 URIs

URL encoding

3.3.2 Peticiones y respuestas

3.3.3 Métodos: GET y POST

3.3.4 Cabeceras HTTP

3.3.5 Códigos de estado

3.3.6 Experimentos

`telnet`

`netcat`

`curl`

`http`

Google Chrome Developer Tools

3.3.7 Envío de datos al servidor

Mediante GET

Mediante POST

Formularios HTML

3.3.8 Cookies

3.4 Apache básico

3.4.1 Instalación

3.4.2 Configuración básica

3.4.3 Sitios virtuales

3.5 Scripts CGI

3.5.1 Configuración de Apache

3.5.2 Ejemplos en Ruby

CAPÍTULO 4

Sistemas de control de versiones

4.1 Git básico

1. config, init, add, commit
2. status, log, diff
3. Alias lg
4. checkout, reset, revert, –amend
5. show
6. rm, mv
7. Atom y Git

4.2 Git avanzado

1. Ramas: branch
2. merge, rebase
3. Resolución de conflictos

4.3 Ramas remotas

1. clone, fetch, push, pull
2. Ramas de seguimiento (tracking branch)

4.4 Repositorios de código (Github.com, GitLab.com, Bitbucket.org...)

Parte II

PHP

CAPÍTULO 5

Conceptos básicos de PHP I

ricpelo's note: Programada inicialmente para empezar el 23-10-2017.

5.1 Introducción a PHP

5.1.1 Página web de PHP

<http://php.net>

5.1.2 Instalación de PHP

5.1.3 Documentación y búsqueda de información

5.1.4 Configuración básica con `php.ini`

```
error_reporting = E_ALL

display_errors = On

display_startup_errors = On

date.timezone = 'UTC'
```

5.2 Sintaxis básica

<http://php.net/manual/es/language.basic-syntax.php>

5.2.1 Datos e instrucciones

En todo lenguaje de programación existen dos elementos básicos: los **datos** y las **instrucciones**. Los datos son las porciones de información con las que trabajan los programas, y las instrucciones son las acciones que manipulan esos datos para llevar a cabo la tarea para la que fue concebido el programa. Ambos elementos, datos e instrucciones, constituyen los pilares del lenguaje y de los programas que se escriben con él.

Desde un punto de vista sintáctico, en el código fuente del programa, los datos se codifican en forma de **expresiones**, y las instrucciones toman la forma de **sentencias**.

La diferencia fundamental entre una expresión y una sentencia es la siguiente:

- Una expresión tiene un valor (se dice que *denota* o *representa* un valor), y por eso decimos que una expresión *se puede evaluar*, y al evaluarse, se obtiene el valor de la expresión, que no es más que un dato.
- En cambio, una sentencia no denota ningún valor, por lo que no puede evaluarse. El intérprete *ejecuta* la sentencia y se llevan a cabo las acciones que provoca dicha ejecución.

Las expresiones se evalúan. Las sentencias se ejecutan.

Ejemplos de expresiones

```
5 * (3 + 6)
```

Es una expresión aritmética que involucra números enteros y cuyo valor es `45`.

```
1.6 + 2.3
```

Es una expresión aritmética que involucra números reales y cuyo valor es `3.9`.

```
74
```

Es una constante literal numérica cuyo valor es, precisamente, `74`.

5.2.2 Sentencias y comandos

Comando `echo`

<http://php.net/manual/es/function.echo.php>

5.2.3 Expresiones, operadores y funciones

ricpelo's note: *Ejemplos*: aritmética, `cos()`, `max()`

5.3 Funcionamiento del intérprete

<http://php.net/manual/es/language.basic-syntax.phpmode.php>

5.3.1 Modo dual de operación

ricpelo's note: Se llaman *modo HTML* y *modo PHP*.

5.3.2 Etiquetas <?php y ?>

5.4 Intérprete interactivo

5.4.1 `php -a`

5.4.2 PsySH

<http://psysh.org/>

5.5 Variables

<http://php.net/manual/es/language.variables.php>

5.5.1 Conceptos básicos

<http://php.net/manual/es/language.variables.basics.php>

5.5.2 Destrucción de variables

<http://php.net/manual/es/function.unset.php>

5.5.3 Operadores de asignación por valor y por referencia

<http://php.net/manual/es/language.operators.assignment.php>

ricpelo's note: En `$b =& $a;`, `$b` **NO** está apuntando a `$a` o viceversa. Ambos apuntan al mismo lugar.
<http://php.net/manual/es/language.references.whatdo.php>

5.5.4 Variables predefinidas

<http://php.net/manual/es/reserved.variables.php>

ricpelo's note: `$_ENV` no funciona en la instalación actual (ver `variables_order` en `php.ini`. Habría que usar `get_env()`.

5.6 Tipos básicos de datos

<http://php.net/manual/es/language.types.intro.php>

5.6.1 Lógicos (bool)

<http://php.net/manual/es/language.types.boolean.php>

ricpelo's note: Se escriben en minúscula: `false` y `true`.
<https://github.com/yiisoft/yii2/blob/master/docs/internals/core-code-style.md#51-types>
ricpelo's note: `boolean` es sinónimo de `bool`, pero debería usarse `bool`.

Operadores lógicos

<http://php.net/manual/es/language.operators.logical.php>

ricpelo's note: *Cuidado*:
- `false and (true && print('hola'))` no imprime nada y devuelve `false`, por lo que **el código va en cortocircuito y se evalúa de izquierda a derecha** incluso aunque el `&&` y los paréntesis tengan más prioridad que el `and`.
- Otra forma de verlo es comprobar que `print('uno') and (1 + print('dos'))` escribe `unodos` (y devuelve `true`), por lo que la evaluación de los operandos del `and` se hace de izquierda a derecha aunque el `+` tenga más prioridad (y encima vaya entre paréntesis).
- En el manual de PHP[1] se dice que: *"La precedencia y asociatividad de los operadores solamente determinan cómo se agrupan las expresiones, no especifican un orden de evaluación. PHP no especifica (en general) el orden en que se evalúa una expresión y se debería evitar el código que se asume un orden específico de evaluación, ya que el comportamiento puede cambiar entre versiones de PHP o dependiendo de código circundante."*
- Pregunta que hice al respecto en StackOverflow[2].

5.6.2 Numéricos

Enteros (int)

<http://php.net/manual/es/language.types.integer.php>

ricpelo's note: `integer` es sinónimo de `int`, pero debería usarse `int`.

Números en coma flotante (float)

<http://php.net/manual/es/language.types.float.php>

ricpelo's note: `double` es sinónimo de `float`, pero debería usarse `float`.

Operadores

[1] http://php.net/manual/es/language.operators.precedence.php
[2] https://stackoverflow.com/questions/46861563/false-and-true-printhi

Operadores aritméticos

<http://php.net/manual/es/language.operators.arithmetic.php>

Operadores de incremento/decremento

<http://php.net/manual/es/language.operators.increment.php>

5.6.3 Cadenas (`string`)

<http://php.net/manual/es/language.types.string.php>

ricpelo's note: Se usa `{$var}` y no `${var}` <https://github.com/yiisoft/yii2/blob/master/docs/internals/core-code-style.md#variable-substitution>

Operadores de cadenas

<http://php.net/manual/es/language.operators.string.php>

Concatenación

Acceso y modificación por caracteres

<http://php.net/manual/es/language.types.string.php#language.types.string.substr>

ricpelo's note: - `echo $a[3]`
- `$a[3] = 'x';`

Operadores de incremento/decremento

<http://php.net/manual/es/language.operators.increment.php>

Funciones de manejo de cadenas

<http://php.net/ref.strings>

Extensión *mbstring*

<http://php.net/manual/en/book.mbstring.php>

ricpelo's note: - `$a[3]` equivale a `mb_substr($a, 3, 1)`
- `$a[3] = 'x';` no tiene equivalencia directa. Se podría hacer:

```
$a = mb_substr($a, 2, 1) . 'x' . mb_substr($a, 4);
```

5.6.4 Nulo

<http://php.net/manual/es/language.types.null.php>

ricpelo's note: `is_null()` vs. `=== null` <https://phpbestpractices.org/#checking-for-null>
ricpelo's note: El tipo `null` y el valor `null` se escriben en minúscula.
<https://github.com/yiisoft/yii2/blob/master/docs/internals/core-code-style.md#51-types>

5.6.5 Precedencia de operadores

<http://php.net/manual/es/language.operators.precedence.php>

5.6.6 Operadores de asignación compuesta

ricpelo's note: `$x`` *<op>*\ ``= $y`

5.6.7 Comprobaciones

De tipos

`gettype()`

<http://php.net/manual/en/function.gettype.php>

`is_*()`

<http://php.net/manual/es/ref.var.php>

ricpelo's note: Poco útiles en formularios, ya que sólo se reciben `strings`.

De valores

`is_numeric()`

<http://php.net/manual/es/function.is-numeric.php>

`ctype_*()`

<http://php.net/manual/es/book.ctype.php>

5.6.8 Conversiones

<http://php.net/manual/es/language.types.type-juggling.php>

Coerción, moldeado, forzado o *casting*

<http://php.net/manual/es/language.types.type-juggling.php#language.types.typecasting>

ricpelo's note: Conversión de cadena a número

Conversión a `bool`

<http://php.net/manual/es/language.types.boolean.php#language.types.boolean.casting>

Conversión a `int`

<http://php.net/manual/es/language.types.integer.php#language.types.integer.casting>

Conversión a `float`

<http://php.net/manual/es/language.types.float.php#language.types.float.casting>

Conversión de `string` a número

<http://php.net/manual/es/language.types.string.php#language.types.string.conversion>

ricpelo's note: **¡Cuidado!**: La documentación dice que `1 + "pepe"` o `1 + "10 pepe"` funciona, pero en PHP7.1 da un **PHP Warning: A non-numeric value encountered**.

Conversión a `string`

<http://php.net/manual/es/language.types.string.php#language.types.string.casting>

Funciones de obtención de valores

ricpelo's note: Hacen más o menos lo mismo que los *casting* pero con funciones en lugar de con operadores. Puede ser interesante porque las funciones se pueden guardar, usar con *map*, *reduce*, etc.

`intval()`

<http://php.net/manual/es/function.intval.php>

`floatval()`

<http://php.net/manual/es/function.floatval.php>

`strval()`

<http://php.net/manual/es/function.strval.php>

`boolval()`

<http://php.net/manual/es/function.boolval.php>

Funciones de formateado numérico

`number_format()`

<http://php.net/manual/es/function.number-format.php>

`money_format()`

<http://php.net/manual/es/function.money-format.php>

`setlocale()`

<http://php.net/manual/es/function.setlocale.php>

ricpelo's note: `setlocale(LC_ALL, 'es_ES.UTF-8'); // Hay que poner el *locale* completo, con la codificación y todo (.UTF-8)`

5.6.9 Comparaciones

Operadores de comparación

<http://php.net/manual/es/language.operators.comparison.php>

== vs. ===

Ternario (?:)

<http://php.net/manual/es/language.operators.comparison.php#language.operators.comparison.ternary>

Fusión de null (??)

<https://wiki.php.net/rfc/isset_ternary>

ricpelo's note: Equivalente al `COALESCE()` de SQL.

Reglas de comparación de tipos

<http://php.net/manual/es/types.comparisons.php>

ricpelo's note: `"250" < "27"` devuelve `false`

5.7 Constantes

<http://php.net/manual/es/language.constants.syntax.php>

ricpelo's note: Diferencias entre constantes y variables:
- Las constantes no llevan el signo dólar (`$`) como prefijo.
- Antes de PHP 5.3, las constantes solo podían ser definidas usando la función `define()` y no por simple asignación.
- Las constantes pueden ser definidas y accedidas desde cualquier sitio sin importar las reglas de acceso de variables.
- Las constantes no pueden ser redefinidas o eliminadas una vez se han definido.
- Las constantes podrían evaluarse como valores escalares. A partir de PHP 5.6 es posible definir una constante de array con la palabra reservada `const`, y, a partir de PHP 7, las constantes de array también se pueden definir con `define()`. Se pueden utilizar arrays en expresiones escalares constantes (por ejemplo, `const FOO = array(1,2,3)[0];`), aunque el resultado final debe ser un valor de un tipo permitido.

5.7.1 `define()` y `const`

5.7.2 Constantes predefinidas

<http://php.net/manual/es/language.constants.predefined.php>

5.7.3 `defined()`

<http://php.net/manual/es/function.defined.php>

5.8 Flujo de control

5.8.1 Estructuras de control

<http://php.net/manual/es/language.control-structures.php>

Sintaxis alternativa

<http://php.net/manual/es/control-structures.alternative-syntax.php>

ricpelo's note: El `do { ... } while (...);` **no** tiene sintaxis alternativa.

5.8.2 Inclusión de archivos

`include`, `require`

<http://php.net/manual/es/function.include.php>

ricpelo's note: El nombre del archivo debe aparecer con su extensión. No vale hacer `require 'pepe';`.
ricpelo's note: Cuando un archivo es incluido, el intérprete abandona el modo PHP e ingresa al modo HTML al comienzo del archivo objetivo y se reanuda de nuevo al final.
ricpelo's note: Si el archivo incluido tiene un `return ...;`, el `include` o el `require` que lo incluya devolverá el valor devuelto por el `return`.

include_once, require_once

<http://php.net/manual/es/function.include-once.php>

5.9 Funciones predefinidas destacadas

5.9.1 isset()

<http://php.net/manual/es/function.isset.php>

ricpelo's note: Cuidado si la variable contiene `null`.
ricpelo's note: No da error ni advertencia si la variable no existe.

5.9.2 empty()

<http://php.net/manual/es/function.empty.php>

ricpelo's note: Para evitar el problema de `empty("0") === true`:

```
function is_blank($value) {
    return empty($value) && !is_numeric($value);
}
```

ricpelo's note: No da error ni advertencia si la variable no existe.

5.9.3 var_dump()

<http://php.net/manual/es/function.var-dump.php>

5.10 Arrays

<http://php.net/manual/es/language.types.array.php>

ricpelo's note: Las claves pueden ser enteros o cadenas.

5.10.1 Operadores para arrays

<http://php.net/manual/es/language.operators.array.php>

ricpelo's note: **Comparaciones**: Un `array` con menos elementos es menor. De otra forma, compara valor por valor.

Acceso, modificación y agregación

<http://php.net/manual/es/language.types.array.php#language.types.array.syntax.modifying>

5.10.2 Funciones de manejo de arrays]

<http://php.net/manual/es/book.array.php> <http://php.net/manual/es/ref.array.php>

Ordenación de arrays

<http://php.net/manual/es/array.sorting.php>

`print_r()`

`'+'` vs. `array_merge()`

`isset()` vs. `array_key_exists()`

<http://php.net/manual/es/function.array-key-exists.php#107786>

5.10.3 `foreach`

<http://php.net/manual/es/control-structures.foreach.php>

5.10.4 Conversión a `array`

<http://php.net/manual/es/language.types.array.php#language.types.array.casting>

5.10.5 *Ejemplo*: `$argv` en CLI

<http://php.net/manual/es/reserved.variables.argv.php>

5.11 Funciones definidas por el usuario

<http://php.net/manual/es/language.functions.php>

5.11.1 Argumentos

<http://php.net/manual/es/functions.arguments.php>

Paso de argumentos por valor y por referencia

<http://php.net/manual/es/functions.arguments.php#functions.arguments.by-reference>

Argumentos por defecto

<http://php.net/manual/es/functions.arguments.php#functions.arguments.default>

ricpelo's note: `php function prueba($opciones = []) { extract($opciones); // ... }`

5.11.2 Ámbito de variables

<http://php.net/language.variables.scope>

Ámbito simple al archivo

Variables locales

Uso de `global`

ricpelo's note: Usar `global $x;` cuando `$x` no existe hace que `$x` empiece a existir y valga `null`.

Variables superglobales

<http://php.net/manual/es/language.variables.superglobals.php>

5.11.3 Declaraciones de tipos

ricpelo's note: **NO** se hacen conversiones implícitas a `array`, ni en argumentos ni en devolución.

Declaraciones de tipo de argumento

<http://php.net/manual/es/functions.arguments.php#functions.arguments.type-declaration>

Declaraciones de tipo de devolución

<http://php.net/manual/es/functions.returning-values.php#functions.returning-values.type-declaration>

Tipos *nullable* (?) y `void`

<http://php.net/manual/es/migration71.new-features.php>

Tipificación estricta

<http://php.net/manual/es/functions.arguments.php#functions.arguments.type-declaration.strict>

ricpelo's note: El `declare(strict_types=1);` se pone en el archivo que hace la llamada, no en el que define la función.

Parte III

Yii2

CAPÍTULO 6

Introducción a Yii2

CAPÍTULO 7

Conceptos fundamentales de Yii2

CAPÍTULO 8

Estructura de una aplicación Yii2

CAPÍTULO 9

Gestión de peticiones en Yii2

CAPÍTULO 10

Acceso a bases de datos en Yii2

CAPÍTULO 11

Recogida de datos y validación de formularios

CAPÍTULO 12

Visualización de datos en Yii2

CAPÍTULO 13

Características adicionales de Yii2

CAPÍTULO 14

Seguridad y cacheado en Yii2

CAPÍTULO 15

Pruebas, documentación y mantenimiento

CAPÍTULO 16

Computación en la nube

CAPÍTULO 17

Contenedores

Parte IV

Índices y tablas

- genindex
- search