PHP PANDAS

IL PHP PER TUTTI

DAYLE REES & FRANCESCO MALATESTA

PHP Pandas (IT)

The PHP Programming Language for Everyone.

Dayle Rees and Francesco Malatesta

This book is for sale at http://leanpub.com/php-pandas-it

This version was published on 2015-11-17

Indice

Ringraziamenti

Innanzitutto vorrei ringraziare la mia ragazzza Emma. Non solo per assecondarmi in tutti i miei capricci piÃ¹ nerd, ma anche per aver scattato la fantastica foto che vedi in copertina. Ti amo, Emma!

Un enorme grazie va anche ai miei genitori, che mi hanno sempre sostenuto negli anni. A loro va anche un grazie per aver comprato un miliardo di copie del libro da distribuire ai vari familiari!

Vorrei anche ringraziare i miei meravigliosi colleghi a JustPark, per il loro supporto costante! Ragazzi, spaccate davvero!

Grazie anche a tutti quelli che hanno comprato, negli ultimi anni, Code Happy e Code Bright. E a tutta al community di Laravel: senza il vostro supporto non avrei mai neanche pensato di avere la capacitÃ (ed il coraggio) di continuare a scrivere.

Errata

Questo che leggi è il mio terzo libro. Probabilmente rispetto al primo sono migliorato un po', ma comunque ho ancora tanto da imparare. Sono sicuro, ad ogni modo, che qui in giro ci sono ancora un po' di errori.

Se dovessi trovarli, e se ti va, segnalameli con una mail completa di tutto quello che ritieni necessario a me@daylerees.com[1], insieme al titolo del capitolo per permettermi di poter "operare" più velocemente.

Gli errori verranno sistemati mano a mano, e le fix saranno pubblicate in futuro, in occasione dei vari aggiornamenti del libro.

[1] mailto:me@daylerees.com

Feedback

Naturalmente, ogni tipo di feedback è ben accetto. Se ti va di suggerire qualcosa, scrivimi all'indirizzo me@daylerees.com[2] oppure usa twitter: sono @daylerees.

Cercherò di rispondere il prima possibile ad ogni singolo messaggio.

[2] mailto:me@daylerees.com

Traduzioni

Vuoi tradurre PHPandas nella tua lingua? Scrivimi a me@daylerees.com[3]. Divideremo a metà i ricavati della traduzione. Il prezzo di vendita sarà lo stesso dell'edizione inglese.

Una nota tecnica: **il libro è stato scritto nel formato Markdown**. Assicurati di conoscerlo prima di partecipare.

[3] mailto:me@daylerees.com

1. Introduzione

Ehilà! Ma chi abbiamo qui? Se non sbaglio il più bel lettore che il nostro pianeta abbia mai visto! Beh, grazie per aver comprato PHPandas. Ti faccio le mie congratulazioni: stai per muovere i primi passi verso una grandiosa carriera da sviluppatore web.

Chi sono io? Beh, semplice: mi chiamo Dayle, sono l'autore di questo libro e dell'avventura in cui ti addentrerai oggi. Da qualche anno scrivo libri per principianti nella programmazione e ho avuto la fortuna di insegnare a tanti "nuovi arrivati" un po' di cose molto interessanti. Anche stavolta sarà così, o almeno me lo auguro. Ovviamente non ti preoccupare: non ti lascerò mai solo.

> Uhm… perché scrivi come uno svitato?

Prego? Oh, si. Come spiegartelo… diciamo che questo è l'unico modo di scrivere che conosco. Se cerchi un manuale tecnico pieno zeppo di termini tecnici e formalità, probabilmente questo non è quello giusto per te. Mi piace scrivere libri per persone. E mi piace parlarci, con le persone. Un po' come con degli amici, di sera, al pub.

Inoltre, ho scoperto con immenso piacere che molti "nuovi arrivati" tendono ad apprezzare questo stile di scrittura. Quando inizi non cerchi di certo una testo specialistico: tutto quello di cui hai bisogno è capire alcuni concetti base! E fidati: è esattamente quello a cui arriverai.

Promesso!

Poi, ci hai fatto caso? In realtà stiamo già parlando, come se fossimo faccia a faccia. Diciamo che questo è il mio potere: riesco ad intuire quelle che saranno le domande che mi farai ed io ti risponderò!

> Cavolo, come diamine hai fatto a…

Beh, è un segreto. Al momento non posso rivelare nulla di più. Informazioni classificate, sai… ad ogni modo, non è meglio essere parte di un'avventura, al posto di esserne solo un osservatore?

> Penso di si… ok, hai la mia attenzione!

Eccellente.

Ora: questo, in qualsiasi altro libro, è il momento in cui ti viene spiegata qualche nozione in più sul PHP. Storia, applicazioni, l'autore ed un milione di altre cose più o meno importanti e che, più o meno, ricorderai. Come ti ho già spiegato non sono l'autore più "tradizionale" che incontrerai, e a questo genere di capitoli non sono particolarmente affezionato. Se hai comprato questo libro è normale che tu sia interessato al PHP, per cui penso tu abbia già letto qualche piccola curiosità sul linguaggio.

Facciamo un riassunto veloce: PHP è un linguaggio di programmazione che è alla base di molte, moltissime realtà su internet. Il suo autore principale, che ne ha curato l'inizio, è Rasmus Lerdorf. Può essere visto sorridere su Google, se lo cerchi in "Immagini". Rasmus è un tizio davvero grande, e penso dovremmo ringraziarlo tutti per quello che è riuscito a creare.

Arrivati a questo punto, molti altri libri sul PHP si occuperebbero di spiegarti quali sono i cereali preferiti di Rasmus. Non questo: facciamo un passo avanti e cerchiamo di trovare la risposta ad una curiosità molto più importante.

Cosa stiamo per imparare?

Chiariamo una cosa: questo libro è innanzitutto per i ***principianti***. Anche quelli più assoluti, che nella vita non hanno mai provato prima d'ora la programmazione. Se hai già tentato qualcosa, comunque, non dovresti avere problemi. Nel caso in cui, invece, tu sia un programmatore più esperto, puoi sempre usare questo libro per ripassare le basi, di tanto in tanto.

Per assicurarmi il livello di comprensibilità degli argomenti, ho usato la mia ragazza ed alcuni miei colleghi (non tecnici) come cavie da laboratorio per vedere in prima persona la loro reazione al libro. Non solo sono ancora vivi, ma hanno reagito perfettamente. Ragion per cui anche tu non dovresti avere problemi.

Il mio obiettivo per questo libro è semplice ed ambizioso: diventare il libro sul PHP più leggero e divertente sul mercato. Mi piacerebbe vedere, un giorno, questo libro come *il libro suggerito* per iniziare a guardare un po' nel mondo PHP e vedere cosa c'è. Gran parte del lavoro, infatti, è stata proprio cercare di ottimizzare i contenuti per renderli il più possibile accessibili. D'altronde, è ben saputo che chi comincia è a metà dell'opera!

Una volta finito, se dovesse piacerti, consiglialo ai tuoi amici!

Ci tengo, inoltre, che tu sappia che questo libro non ti spiegherà come creare dei siti con PHP (ci sarà presto un altro volume appositamente concepito per tale scopo), ma si concentrerà sugli aspetti più essenziali del linguaggio. Immaginalo come un primo passo verso una conoscenza più ampia.

L’inizio.

Come già detto nella sezione feedback, per qualsiasi richiesta/segnalazione ed altro usa pure il mio indirizzo email `me@daylerees.com`!

Detto questo, non sprechiamo altro tempo. Hai un bel po’ di cose da imparare! Gira la pagina ed immagina il cancello di Jurassic Park che si apre verso un nuovo ed incredibile mondo.

2. Installazione

Prima di iniziare a lavorare con PHP dobbiamo innanzitutto installarlo. Come probabilmente immagini già, PHP è un'applicazione come tante altre: deve essere installata sul sistema prima di poter processare del codice.

Ora, il metodo di installazione varia in base al sistema operativo che stai usando. Per questo motivo, ho deciso di scrivere tre guide differenti. La prima spiegherà come installare PHP su una distribuzione Linux.

Userò Ubuntu come esempio per via della sua popoloarità. La seconda parlerà della stessa operazione ma su Mac OSX, mentre la terza riguarderà Windows.

Visto e considerato che NON creeremo siti web, installeremo **solo la versione console** di PHP.

Leggi solo la sezione relativa al tuo sistema operativo. Salta pure quello che non ti serve e poi via, verso il prossimo capitolo!

Linux

Il miglior modo di installare PHP su una distribuzione basata su Linux è tramite il package manager. Il package manager varia in base alla distribuzione scelta. In questo caso, per via della sua popolarità, ho deciso di mostrarti la procedura di installazione su Ubuntu. Ubuntu usa il package manager `apt` ed il package da installare è `php5-cli`.

Tutto quello che dovrai fare sarà scrivere la seguente istruzione nella console.

```
1  $ sudo apt-get install php5-cli
```

Occhio: non ci sarà bisogno di scrivere anche quel simbolo del dollaro all'inizio: indica semplicemente che stiamo lavorando con la console.

Una volta premuto invio per dare l'ok, il package verrà recuperato ed installato. Fine.

Semplice, vero? Per sicurezza, verifichiamo effettivamente se PHP è stato installato.

```
$ php -v
```

Questo comando viene spesso usato come verifica: mostra la versione attualmente installata di PHP. Dovresti vedere qualcosa del genere, come output:

```
PHP 5.5.13 (cli) (built: Jun  5 2014 19:13:23)
Copyright (c) 1997-2014 The PHP Group
Zend Engine v2.5.0, Copyright (c) 1998-2014 Zend Technologies
```

Quello che vedrai non sarà esattamente quello che leggi qui sopra, adesso.

D'altronde il mondo è bello perché è vario, no?

Scherzi a parte, PHP viene costantemente migliorato e le versioni cambiano. Sicuramente, nel momento in cui leggerai queste parole, la versione disponibile sarà sicuramente oltre la `5.4.0`.

Niente di più! Passa pure al prossimo capitolo, qui abbiamo finito!

Mac OSX

Sui sistemi operativi Macintosh, PHP è già installato di default. Per cui, tutto quello che devi fare è eventualmente controllare la versione installata tramite il comando (da inserire nella console)

```
$ php -v
```

Non digitare anche il simbolo del dollaro, rappresenta semplicemente il fatto che stai lavorando con la console. L'output che vedra sarà simile al seguente.

```
PHP 5.4.24 (cli) (built: Jan 19 2014 21:32:15)
Copyright (c) 1997-2013 The PHP Group
Zend Engine v2.4.0, Copyright (c) 1998-2013 Zend Technologies
```

Nell'esempio qui sopra la versione è la `5.4.24`. Fin quando questo numero è al di sopra di `5.4` nessun problema, puoi passare al capitolo successivo.

Altrimenti bisogna usare un package manager di terze parti per installare la nuova versione di PHP. Niente di complesso, quindi non farti spaventare: vediamo come fare.

Useremo un package manager chiamato 'Homebrew', per gli amici 'Brew'. Installarlo è semplice: tutto quello che devi fare è seguire le istruzioni che troverai sul sito

brew.sh[1]

Una volta installato Brew, tutto quello che dovrai fare sarà usare un semplice comando per effettuare l'installazione di PHP. Così:

```
$ brew install php55
```

Infine, aggiungi il percorso di PHP alla variabile di sistema PATH. Così:

```
$ PATH=~/usr/local/Cellar/php55/5.5.13/bin:$PATH
```

Se dovessi avere problemi, controlla che la versione inserita nella stringa combaci con quella presente nel tuo sistema. A questo punto non ti rimane che controllare l'effettivo aggiornamento tramite

```
$ php -v
```

Se questa volta la versione è maggiore di `5.4` ci siamo! Puoi procedere verso il prossimo capitolo.

Windows

Installare PHP su Windows è leggermente più difficile rispetto agli altri sistemi operativi. Ho testato il metodo anche su Windows 10: non dovresti avere problemi. In caso contrario fammi sapere e cercherò di scoprire qualcosa di più a riguardo.

Innanzitutto, punta all'indirizzo http://windows.php.net/download[2]. Da qui scarica l'ultima versione di PHP, come archivio zip. Una volta scaricato l'archivio devi estrarlo in una cartella adeguata. Nel mio caso ho messo tutto in

```
C:\Users\Dayle\PHP
```

A questo punto avrai bisogno di un prompt per eseguire i vari script che scriveremo in questo libro. Per comodità, ecco cosa puoi fare per averne uno sempre a disposizione.

Sul desktop, clicca con il pulsante destro del mouse su una zona "vuota" e scegli "Crea scorciatoia". Come destinazione del collegamento inserisci semplicemente

[1] http://brew.sh/

[2] http://windows.php.net/download

```
cmd.exe
```

Clicca su “Avanti” e dai alla scorciatoia appena creata il nome “PHP”.

Infine, clicca con il pulsante destro del mouse sulla scorciatoia e seleziona “Proprietà”. Sulla tab “Scorciatoia” cambia il campo “Esegui in” facendo in modo che corrisponda al percorso dove hai estratto l’archivio. Clicca su OK una volta terminata la procedura.

Apri quindi il tuo prompt e scrivi:

```
php -v
```

... che dovrebbe mostrarti la versione attuale di PHP installata sul tuo sistema.

Fatto! A questo punto puoi andare avanti.

3. Trovare le Risposte

Lo so. Sembra un po' moscio e vago come titolo, non è vero? Beh, fidati se ti dico che qui parleremo di cose davvero importanti. Forse di una delle cose più importanti in assoluto. Questo capitolo è interamente dedicato a quella che sarà la "confidenza" di cui avrai bisogno, nel tuo percorso, per diventare uno sviluppatore.

Imparare è difficile, ma non ti preoccupare. Sono qui anche per questo.

Gli Sviluppatori sono dei Robot

Perché hai deciso di seguire questa strada? Non dirmi niente, aspetta! Fammi indovinare. Scommetto che hai visto uno sviluppatore PHP, uno di quelli eccezionali, leggendari, uscire da una limousine durante una di quelle notti selvagge a New York, ordinare al volo 5 bottiglie di Cristal e rilassarsi un po' con Jay-Z ed il fantasma di Tupac.

Vero, la vita di uno sviluppatore è una delle più glamour a cui una persona può ambire. Io stesso sto scrivendo questi capitoli durante le mie sole cinque ore di sobrietà che mi concedo ogni giorno.

Oppure, semplicemente, hai visto uno sviluppatore scrivere del codice, una volta, e hai pensato…

> Diamine, questo tizio deve essere un robot! Conosce a memoria tutto il codice che deve scrivere, tutte quelle parentesi, tutti quei caratteri strani… Wow!

Quando le persone "normali" hanno a che fare con uno sviluppatore pensano di avere davanti un genio in matematica con diciotto lauree. Si, per alcuni sviluppatori è vero, ma non di certo nel mio caso (e probabilmente anche in altri). Molti, suppongo, saranno d'accordo con me.

La verità è diversa: nessuno di noi è perfetto, neanche un minimo. Se pensi che uno sviluppatore conosca a memoria tutte quelle funzioni e tutte quelle parole chiave, probabilmente stai anche pensando che non sarai mai a "quel livello".

Niente di più sbagliato.

Semplicemente, **non è vero**. Non memorizziamo tutto.

In realtà, la maggior parte del codice che usiamo tutti i giorni viene da una reference, o da una documentazione. Siamo, in un certo senso, dei guerrieri la cui arma principale è Google. Si, sappiamo usare questa nostra arma in modo eccezionale.

Ad esempio, ci sono delle funzioni per le quali non ricordo la sintassi. Uso Google. Non ricordo una particolare procedura? Uso Google. Molto spesso, altri sviluppatori hanno affrontato il mio problema prima di me, e risolto meglio di come l'avrei potuto fare io, da solo.

Chiaramente, la cosa funziona a doppio senso: cerco di aiutare la community come posso, rispondendo sui forum, postando le mie risposte alle domande su Stack Overflow e così via.

Ecco, questa è una delle cose più belle ed importanti per uno sviluppatore. Dare indietro alla community quello che ti ha prestato. Uno scambio equo, di conoscenze.

No, non siamo robot. Non conosciamo tutto di un linguaggio e no, non avremo mai una soluzione per ogni problema. Tuttavia, siamo ottimi ricercatori e questo possiamo dirlo. Siamo opportunisti e dei risolutori di problemi pieni di risorse.

Siamo sviluppatori.

L'Elegante Arte del Googlare

Quando le persone ti dicono di "googlare" qualcosa, è davvero semplice prendersela come fosse un insulto. O magari sarcasmo? In realtà, non ricordo... comunque. Google è molto spesso la nostra home page: lo è per una buona ragione. Vediamo come fare per trovare delle risposte a problemi riguardanti lo sviluppo piuttosto comuni.

Chiudi gli occhi e seguimi. Metaforicamente intendo, perché devi continuare a leggere.

Stiamo scrivendo un programma. Immaginiamo che questo programma debba prendere una frase e fare in modo di stampare su schermo l'inversa. Ovvero una stringa con i caratteri in ordine inverso rispetto a quella di partenza. Da 'Pandas rule!' a '!elur sadnaP', per capirci. Non sappiamo come fare, e stiamo iniziando proprio ora ad imparare PHP.

Ora, sappiamo che in PHP una sequenza di testo è detta stringa (string in inglese). Come? Beh, perché non ci siamo fermati a questo capitolo e l'abbiamo scoperto in uno dei prossimi capitoli.

Vogliamo fare in modo di invertire l'ordine dei caratteri in una stringa. Si, ma come?

Google è il nostro migliore amico e la nostra migliore arma: tutto sta nel costruire la giusta richiesta di ricerca. In gergo tecnico viene detta "query di ricerca".

Ad esempio:

```
1 reverse string
```

Semplice, no? Ah, quando fai ricerche inerenti la programmazione ti suggerisco di usare l'inglese. Nella programmazione è indispensabile conoscerlo e se non lo conosci è un ottimo motivo per impararlo.

Aspetta però, fermo! Ci sono decine di linguaggi di programmazione in giro. Molto probabilmente, inserendo come query di ricerca il semplice "reverse string" otterresti un sacco di risultati per un sacco di linguaggi! C++, ASP.NET, Erlang e così via. A noi tutto questo non interessa, perché abbiamo bisogno della soluzione per PHP!

Quindi, cambiamo la query da "reverse string" a...

```
1 php reverse string
```

Perfetto. Vediamo adesso che cosa otteniamo come risultato.

Prima di andare avanti una piccola precisazione: non lavoro per Google, e non mi pagano nessuna commissione per menzionarlo nell'esempio che stai leggendo! Se vuoi, sentiti libero di usare Bing, ma probabilmente finiresti per ritrovarti a comprare un rimorchio da cavallo usato (il rimorchio, non il cavallo, o forse anche quello) al posto di trovare quello che cerchi.

Comunque, ecco cosa restituisce la ricerca.

Reverse a string - PHP

http://www.php.net/manual/en/function.strrev.php[1]

Reverse a string with php - Stack Overflow

http://stackoverflow.com/questions/11100634/reverse-a-string-with-php[2]

---

[1] www.php.net/manual/en/function.strrev.php

[2] http://stackoverflow.com/questions/11100634/reverse-a-string-with-php

La giusta domanda molto spesso porta alla giusta risposta senza troppi giri. Il **Manuale PHP** (spesso conosciuto anche come “PHP API Docs”) e **Stack Overflow** sono due delle risorse più utili nel mondo PHP. Le userai molto spesso. Non dico che hanno sempre la soluzione giusta per qualsiasi problema, ma sono sicuro che verranno fuori nelle tue ricerche molto spesso.

Comunque, vediamo ora che cosa il buon PHP ci mette a disposizione per invertire i caratteri di una stringa. D’altronde non stiamo risolvendo un problema astratto, sappiamo esattamente quello che vogliamo!

Cliccando sul primo risultato, scoprirai che PHP mette a disposizione una funziona `strrev()`. Non devi sapere cosa fa nel dettaglio e non hai bisogno di vederne il codice. Sai che effettua l’operazione che cerchi. Nulla di più.

Il manuale PHP inoltre ti spiegherà per bene come usare questa funzione, quali sono i parametri da specificare nel suo uso e di come sistemare la sintassi in modo adeguato.

Come ciliegina sulla torta, molto probabilmente ti metterà a disposizione anche un numero sufficiente di esempi a farti capire il suo funzionamento in tutti i casi possibili.

Visto? Non sapevi nulla della funzione `strrev()` fino a poco fa ed ora sai come risolvere il tuo problema.

Magari tra tre mesi non la ricorderai precisamente: la documentazione, però, continuerà ad essere dove l’hai trovata.

Dopo averla usata un po’ di volte saprai già come si chiama, ma non ricorderai i parametri da usare: nessun problema, la documentazione sarà ancora dove l’hai trovata.

La lezione, quindi, quale sarebbe? Semplice: non andare in panico. Non devi necessariamente ricordarti tutto in ogni momento ed è perfettamente naturale aver bisogno d’aiuto. Così com’è umano chiedere aiuto, è umano imparare dalle proprie esperienze.

Congratulazioni! Sei un essere umano. Non un robot.

4. File PHP

Ok, sei pronto per una news scioccante?

> Mmh… no.

Bene: il codice PHP viene messo in un file.

> Ma avevo detto…

Mi spiace dirtelo così, ma è vero. Beh, forse all'inizio lavorerai con un solo file, ma in un secondo momento lavorerai con molti file insieme, probabilmente.

Dopo questa scottante verità, vediamo come creare un file PHP.

> Dayle, guarda che almeno le basi basi le conosco, eh…

Ma che bravo che sei! Buon per te, ma non intendevo questo. In realtà, molti file PHP hanno alcune cose in comune… parlo ad esempio del tag di apertura di un file PHP.

Precisamente, questo.

```
1 <?php
```

Bello, vero? Fantastico e meraviglioso. Una perla, un'ancora di salvataggio in questo mare tempestoso chiamato vita.

> Io… ehm…

Immagino tu stia pensando la stessa identica cosa, vero? Dopo tanti anni di sviluppo PHP troverai tutto questo sempre bellissimo. Lo avrai davanti ai tuoi occhi perfino la sera, prima di andare a dormire.

Scherzi a parte preferisco sempre iniziare con un bell'esempio, per cui vediamo di fare qualcosa insieme. Crea un nuovo file e chiamalo `test.php`. Ogni file PHP ha l'estensione PHP. In realtà puoi eseguirlo anche se non ha l'estensione, ma è buona norma e convenzione usarla. Scriviamoci dentro qualche parola…

```
Pandas rule!
```

... ed infine salviamo il file.

Bene, è arrivato il momento di eseguirlo. Ed è piuttosto semplice farlo, perché basta usare, da linea di comando, l'applicazione PHP e passargli il nome del file come parametro. Ad esempio, da Mac dovrò eseguire il comando

```
php test.php
```

Vedrai le parole `Pandas rule!` sullo schermo. Questo perché **qualsiasi testo al di fuori del tag PHP non viene processato** quando l'applicazione viene eseguita. Proviamo qualcosa di più. Usiamo il nostro primo tag PHP!

Modifica il file, stavolta, così:

```
<?php

        // Pandas are awesome!

?>

Pandas rule!
```

Salva il file ed esegui nuovamente. Cosa succede? Ecco l'output ottenuto.

```
Pandas rule!
```

> Ehi aspetta! Dov'è finito il resto?

Ci hai fatto caso, vedo! Ottimo, caro il mio apprendista. C'è una sezione del nostro file che, in effetti, manca all'appello. Questo perché tutto quello che inserisco all'interno del tag PHP viene trattato come codice. Di conseguenza, viene processato.

Il tag di apertura, `<?php`, l'hai già conosciuto prima. C'è tuttavia anche il tag di chiusura `?>`, che segna la fine dell'area da "interpretare".

Ora che sai come funziona il sistema dei tag in PHP, è facile capire dove si trova il codice eseguito in questo file. Precisamente, in questa linea:

```
1  // Pandas are awesome!
```

A questo punto ti starai chiedendo cosa fa questo codice: beh, nulla. Ed è conosciuto, nel mondo PHP, come un “commento”. Non preoccuparti se non sai cos’è: ti spiegherò tutto tra un po’.

Dopo questa prima e doverosa introduzione al tag più bello che c’è, direi che posso darti la buona notizia: nel prossimo capitolo scriverai, finalmente, ***le tue prima vere righe di codice PHP***.

Contento? Beh, perché aspettare! Gira questa pagina, dai.

5. Aritmetica di Base

Sono sicuro che, almeno una volta nella vita, avrai sentito che la programmazione è tutta matematica. Vero?

Bene. Finalmente è arrivato il momento di fare un po' di matematica.

$$\left|\sum_{i=1}^{n} a_i b_i\right| \le \left(\sum_{i=1}^{n} a_i^2\right)^{1/2} \left(\sum_{i=1}^{n} b_i^2\right)^{1/2}$$

Ci sei? Risolvi in X.

Ah ah. Scherzo. In realtà qui non c'è una X, e questa non è neanche un'equazione. Ad essere onesto non so neanche di preciso cosa sia, questa roba. Benvenuto ne mondo reale: non siamo tutti dei geni matematici.

Istruzioni

Ok, che ne dici di provare qualcosa di più "vicino" al mio livello? Ti ho già spiegato come creare un file PHP, quindi conosci la procedura. Creane uno nuovo e chiamalo `math.php`. Il contenuto sarà il seguente.

```
1  <?php
2
3  3 + 3;
4
5  ?>
```

Fermo un secondo. Aspetta. Non avremo bisogno di mandare in output nulla, dopo il codice PHP, quindi... perché mai disturbarsi ad inserire il tag di chiusura? Togliamolo pure!

```
1  <?php
2
3  3 + 3;
```

Molto meglio!

Nel caso in cui la tua matematica abbia qualche lacuna, lascia che ti dia una mano. Tre più tre fa sei.

Ora possiamo andare avanti.

La riga `3 + 3;` è fondamentalmente un'istruzione. Stiamo scrivendo del codice che verrà interpretato da PHP. Fai caso al punto e virgola `;` alla fine dell'istruzione. **In PHP, ogni istruzione finisce con un punto e virgola**.

All'inizio te ne scorderai sempre, successivamente ti ritroverai ad usarlo anche per finire le frasi nella vita di tutti i giorni.

Visto che capisci bene come funzionano le addizioni di base, quale credi sarà il risultato finale di questa operazione?

> Sette e mezzo.

Oh. Ok, vediamo se hai ragione. Apri il terminale ed avvia `php math.php`. Vediamo che succede.

```
[nessun output]
```

Cosa? Niente di niente! Maledizione, sapevo PHP fosse stupido, ma non così tanto.

Scherzo, scherzo. Ho un delizioso senso dell'humour, vero? Ti ci abituerai.

Analizziamo la situazione: è più semplice di quanto si pensi. Perché non abbiamo visto niente su schermo, durante l'esecuzione del nostro programma? Semplice: PHP è molto obbediente e noi gli abbiamo chiesto di fare un calcolo. Non gli abbiamo chiesto di stampare quacosa su schermo o di invocare lo spirito di Odino. Vero?

Modifichiamo il nostro codice, usando l'istruzione `echo`.

```php
<?php

echo 3 + 3;
```

Ci siamo! Piazzando `echo` prima dell'istruzione dovremmo poter vedere il risultato dell'operazione. Proviamo subito ad avviare il programma e...

```
6
```

Woah! Sei! Ah, non è sette e mezzo.

> Era uno scherzo, Dayle. Comunque avrei potuto usare la calcolatrice.

Lo so, lo so, al momento non stiamo affrontando esattamente un argomento di ingegneria aerospaziale. Me ne occuperò in un altro capit… anzi no, ho usato questa battuta già in un altro libro. Diamine, mi serve altro materiale.

Operatori Aritmetici

So bene che `3 + 3` sembra qualcosa di molto semplice ma non ti preoccupare: presto arriveremo a cose più complesse. Adesso, invece, soffermiamoci sulle basi.

Ecco quali sono gli operatori matematici presenti in PHP.

```
1  + Addizione
2  - Sottrazione
3  * Moltiplicazione
4  / Divisione
5  % Modulo
```

Sono ragionevolmente sicuro che tu abbia visto alcuni di questi operatori prima di oggi. I simboli di moltiplicazione e divisione forse sono leggermente diversi da quelli che hai visto a scuola. Sappi che sono comuni a una stragrande maggioranza di linguaggi di programmazione.

L'unico operatore che forse non hai mai visto è il Modulo. Ad ogni modo è piuttosto semplice da spiegare (e da capire). In poche parole, viene usato per calcolare il resto di una divisione. `3 % 2` sarà uguale ad `1`. Se `/` viene usato per calcolare il risultato, `%` serve a calcolare il resto. Uno degli usi più comuni di questo operatore è quello che se ne fa per scoprire se un numero è pari o dispari.

Proviamo adesso a scrivere un'espressione leggermente più complessa.

```
1  <?php
2
3  echo 4 + 3 * 2 / 1;
```

Riesci a calcolare il risultato a mente? Magari non ricordi quale elemento calcolare prima e quale dopo… ad ogni modo, a prescindere da tutto, in matematica sai bene che si usano le parentesi per separare in modo logico i calcoli da fare e dare un ordine di precedenza.

Nella programmazione l'approccio è lo stesso: puoi usare le parentesi tonde per separare i calcoli come ritieni opportuno.

```php
<?php

echo (4 + 3) * (2 / 1);
```

Adesso siamo sicuri che il calcolo di `4 + 3` verrà eseguito per primo, insieme a `2 / 1`. Il risultato sarà quindi moltiplicato.

Infatti, avviando il programma...

```
14
```

Fantastico! Proviamo ora a togliere le parentesi...

```php
<?php

echo 4 + 3 * 2 / 1;
```

... il risutato sarà:

```
10
```

Ben diverso da quello precedente! Tuttavia, cosa è successo? Niente di che: semplicemente, PHP ha un ordine specifico di valutazione degli operatori matematici. Alcune operazioni vengono effettuate prima, altre dopo. Precisamente, ecco l'ordine degli operatori: quello più in alto ha la massima precedenza.

```
* Moltiplicazione
/ Divisione
% Modulo
+ Addizione
- Sottrazione
```

Adesso immagino tu capisca perché viene eseguito, per primo, `3 * 2 = 6`, poi `6 / 1 = 6` ed infine `4 + 6`. Con l'ovvio risultato di `10`.

Insomma, concludendo: usa le parentesi. Diventa tutto molto più chiaro.

Codice Procedurale

Il codice PHP viene interpretato ed eseguito in modo procedurale. Il che vuol dire che ogni singola istruzione viene presa ed interpretata una dopo l'altra. Certo, puoi specificare più di una istruzione per riga, ma non farà differenza. A livello cronologico rimarrà sempre un'istruzione dopo l'altra. Inoltre, è buona norma seguire la regola del "un'istruzione per ogni riga". Non di più.

Vediamo, adesso, di esaminare qualche altro concetto relativo a questo aspetto del PHP.

Partiamo da questo esempio.

```
1  <?php
2
3  echo 2 + 2;
4  echo 3 + 3;
5  echo 4 + 4;
6  echo 5 + 5;
```

Eseguiamolo e...

```
1  46810
```

> FERMO, COSA?

Prima di andare in panico, fermati. Se ci fai caso, non abbiamo specificato di stampare degli spazi tra un'operazione e l'altra. PHP ha semplicemente calcolato i valori (`2 + 2 = 4`, `3 + 3 = 6`...) e li ha stampati consecutivamente.

Facendo stampare anche degli spazi il risultato sarebbe stato...

4 6 8 10

Ricorda che PHP è molto obbediente, fa esattamente quello che gli dici. Soprattutto mettere uno spazio nel codice (senza stamparlo) non equivale di certo all'inserimento di uno spazio nel risultato finale. Guarda questo altro esempio, per capire meglio.

```php
<?php

echo    2          +   2;
echo                   3   +3;
echo 4+4;
echo    5+     5        ;
```

Non solo non è bello da vedere, ma il risultato sarà esattamente lo stesso di prima! Anche qui c'è una buona regola da seguire sempre: **nel codice, ogni parola va separata dalla successiva con uno spazio**.

Chiaramente, non puoi neanche scrivere una cosa del genere...

```php
<?php

echo5 + 5;
```

... visto che PHP si ritroverebbe a cercare un'istruzione `echo5` che non esiste!

Un codice messo tutto su una sola riga, invece, sarebbe piuttosto sgradevole da leggere.

```php
<?php echo 2 + 2; echo 3 + 3; echo 4 + 4; echo 5 + 5;
```

Nota: questo codice che tu vedi funziona. Tuttavia, non è proprio bello da vedere e non troverai mai un bravo sviluppatore che scrive cose del genere.

Certo, poi puoi fare come vuoi. Addirittura un codice come il seguente è ritenuto "valido" da PHP.

```php
<?php

echo
2
+
2
;
```

Se non ci credi provalo tu stesso: il risultato verrà calcolato e stampato su schermo, ma come vedi non è esattamente un capolavoro di leggibilità!

Tuttavia, c'è un momento in cui andare a capo è molto utile. Immagina di avere un calcolo molto lungo da svolgere. In casi come questi, andare a capo è una buona norma e viene applicata come nell'esempio qui di seguito.

```php
<?php

echo (3 * 5) / (7 / 12) * (7 * 6) + (7 % 3)
        + (6 + 7) * (12 / 3);
```

Se ci fai caso, giusto al di sotto di `echo` sono stati lasciati alcuni spazi. Generalmente quattro spazi (oppure un tab). Adottare una pratica del genere viene vista come un aiuto nella scrittura di codice di qualità. Ricordalo!

Per concludere in bellezza, ricorda anche che le righe vuote non vengono valutate in PHP. Quindi, usale nel modo che ritieni più opportuno.

```php
<?php

echo 3 + 2;

echo 7 * 7;

echo 5;
```

Si, PHP è molto flessibile, ma ci sono alcune convenzioni che è bene fare proprie da subito per migliorare consistentemente, fin dai primi “passi”, la qualità del proprio codice.

Ovviamente, ricorda sempre di aggiungere il punto e virgola alla fine di ogni istruzione;

SEMPRE;

6. Variabili ed Assegnazioni

Iniziamo a giocare con qualcosa di più interessante! Le variabili sono una parte fondamentale, importantissima nel bagaglio di un programmatore. Un concetto del quale non si può fare a meno.

Le “Piccole Scatole”

Si, è il paragone più usato ma anche quello più utile: immagina una variabile come una piccola scatola in cui ci puoi mettere qualcosa. Per non perderla e per poterla riusare in futuro.

In PHP, le variabili sono delle parole che iniziano per `$`. Eccone un esempio:

```php
<?php

$three = 3;
```

Se pensi alla variabile `$three` come una piccola scatola, immagina che al suo interno ci sia il valore `3`. Ecco a cosa serve il simbolo `=` in questa posizione. In matematica a volte lo usiamo per indicare il risultato di un’equazione, ma il concetto non è così differente, alla fine.

Il concetto perno è infatti quello di “assegnazione”, ed infatti `=` è conosciuto come **l’operatore di assegnazione**. Tutto quello che fa infatti è *impostare*, *assegnare* ad una variabile un determinato valore. In questo caso, `3` a `$three`.

Provando ad eseguire un codice del genere, ovviamente, PHP non stamperà nulla su schermo. Esattamente come visto in precedenza, PHP rimane un linguaggio che obbedisce ai tuoi ordini. Usando `echo` le cose cambiano.

```php
<?php

// Imposto il valore 3 per la variabile $three.
$three = 3;

// Stampo il valore della variabile $three.
echo $three;
```

Abbiamo assegnato il valore alla variabile, quindi usato `echo` per mandarne in output il valore. Semplice e lineare. Come concetto è sensazionale, se ci pensi: possiamo dare dei "soprannomi" di comodo a vari valori per poterli riusare in un secondo momento. Anche in modo più agevole. Immagina di dover lavorare ad un programma che gestisce varie operazioni su alcuni cerchi. Sarai d'accordo con me sul fatto che il numero '3.14159265359', nel contesto, ha un certa importanza.

Lo userai molto, molto spesso, per cui... perché non abbreviare il testo da digitare di volta in volta usando qualcosa come:

```
1  <?php
2
3  $pi = 3.14159265359;
```

Così abbiamo creato una variabile `$pi`, che contiene il valore opportuno.

Usare quindi questo valore in altre "zone" del codice sarà molto più semplice.

```
1  <?php
2
3  // Assegno il valore ad una variabile.
4  $pi = 3.14159265359;
5
6  // Calcolo qualche circonferenza.
7  echo $pi * 5;
8  echo $pi * 3;
```

In PHP puoi dichiarare ed assegnare tutte le variabili che vuoi. Ci sono solo alcune regole per quanto riguarda la scelta dei nomi di queste variabili. Ricorda innanzitutto che i nomi delle variabili **possono contenere numeri, lettere ed underscore _**. Inoltre, un nome di variabile **non può mai iniziare con un numero**. Infine, **i nomi sono case sensitive**. La variabile `$pAndA` sarà diversa da `$panda`.

Ecco qualche esempio:

```php
<?php

$panda = 1;     // Valido!
$Panda = 1;     // Valido!
$_panda = 1;    // Valido!
$pan_da = 1;    // Valido!
$pan_d4 = 1;    // Valido!
$pan-da = 1;    // NON Valido!
$4panda = 1;    // NON Valido!
```

Nel dare i nomi alle variabili sono spesso seguite alcune convenzioni, per migliorarne la leggibilità. Non si tratta di regole tassative: il tuo programma continuerà a funzionare, ma è buona norma seguirle per creare un codice di qualità più elevata.

Una di queste "regole" è l'adozione del "camelCasing" nella scelta del nome della variabile.

> Camel... cosa?

In poche parole, quando scegli il nome della variabile fai in modo che la prima lettera sia minuscola. Se ci sono altre parole del nome, di queste altre parole fai in modo che la prima lettera sia maiuscola.

Ecco qualche esempio:

```php
<?php

$laboratorioDiDexter = 1;
$powerRangers = 1;
$spongeBobSquarePants = 1;
```

Come vedi, la prima lettera è sempre minuscola. Le prime lettere delle parole successive sempre maiuscole. Tutto qui!

Detto questo, passiamo avanti e facciamo qualche altro piccolo esempio ed analizziamo altre peculiarità delle variabili in PHP.

```php
<?php

echo $panda = 1337;
```

Mettendo `echo` prima di una procedura di assegnazione possiamo far stampare il valore assegnato subito dopo. Piuttosto comodo, anche se non credo sarà una cosa che userai spesso. Ad ogni modo, fa bene esserne a conoscenza.

Ecco invece un'altra cosa interessante:

```php
<?php

$firstPanda = $secondPanda = $thirdPanda = 1337;
```

Abbiamo effettuato quella che può essere chiamata un'assegnazione a catena. Che si legge così: *il valore 1337 viene assegnato a* `$thirdPanda`. Quindi, *il valore di* `$thirdPanda` *viene assegnato a* `$secondPanda`. Infine, il valore di `$secondPanda` viene assegnato a `$firstPanda`*.

A fine corsa, tutte le variabili hanno lo stesso valore.

Sei il Mio Tipo?

Fino a questo momento abbiamo lavorato essenzialmente con i numeri. Sarebbe piuttosto noioso, però, continuare così ad oltranza. D'altronde, PHP offre svariate possibilità e ci sono molti tipi di variabili da usare per la propria applicazione.

Ecco quelli più comuni:

- integer
- float
- boolean
- string
- null
- array

Nota di Traduzione: ho lasciato i nomi in inglese per familiarizzare meglio con il concetto.

In realtà ce ne sarebbero alcuni in più, ma per ora non complichiamoci la vita. Per il resto di questo capitolo analizzeremo un tipo alla volta.

Iniziamo subito dagli integer, ovvero i numeri interi. Li abbiamo già usati in precedenza, negli esempi.

```php
<?php

$panda = 2;
$redPanda = -23;
```

I float, invece, sono numeri decimali a virgola mobile. Possono essere usati per rappresentare frazioni e contengono, appunto, cifre dopo la virgola. Anche loro li abbiamo già visti: nell'esempio fatto per `$pi`, ricordi?

```php
<?php

$panda = 2.34;
$redPanda = -23.43;
```

Subito dopo abbiamo quindi i booleani. Niente panico, il nome magari ti ricorda una miriade di operazioni noiose con 1 e 0. Non è questo il caso. Semplicemente, una variabile booleana viene usata per esprimere un concetto che normalmente può essere rappresentato con soli due valori. Uno o zero, acceso o spento, vero o falso. I valori che una variabile booleana può assumere sono `true` o `false`.

Successivamente vedremo come possiamo usare le variabili booleane per cambiare l'intero flusso della nostra applicazione. Nel frattempo, ecco qualche altro esempio.

```php
<?php

$panda = false;
$redPanda = true;
```

Arriviamo alla variabile di tipo stringa (string). Tali variabili vengono usate per memorizzare parole, caratteri oppure intere sequenze di testo. Gli usi sono molteplici. Le stringhe sono speciali: a loro dedicherò, a breve, un capitolo separato.

```php
<?php

$panda = 'Normal Panda';
$redPanda = "Red Panda";
```

C'è quindi Null, che è un valore speciale. Il `Null` è il nulla. Il vuoto, l'assenza. Neanche lo zero rende bene l'idea. Lo zero è qualcosa, è un valore. Il Null, semplicemente, non è. In futuro vedremo come usarlo nel dettaglio. A volte dipenderà da te: ci sono persone che non lo usano per principio, altre invece he lo usano senza problemi. Con il tempo ti farai un'idea tua.

```php
<?php

$noPanda = null;
```

Gli array sono un altro tipo "speciale" di variabile. Infatti, un array non è una singola variabile, ma una vera e propria collezione di valori.

```php
<?php

$countThePandas = [1, 2, 3];
$morePandas = array(5, 6, 7, 8);
```

Anche gli array li vedremo in un capitolo separato. Sono il mio tipo di dato preferito e non potevo assolutamente rischiare di lasciarli fuori dal gioco!

Assegnazioni "Avanzate"

Nel capitolo precedente abbiamo scoperto gli operatori da usare sulle variabili. Abbiamo inoltre imparato a conoscere l'operatore di assegnamento. A questo punto la domanda sorge sponteanea...

> Ehi, e se provassi ad unirli?

Perché no? Facciamo qualche prova, dai!

```php
<?php

// Assegno un valore.
$panda = 3;

// Faccio esperimenti, sperando che non si apra un buco nero.
$panda += 1;

// Reggiti forte alla scrivania e vediamo che succede.
var_dump($panda);
```

Come tu stesso vedi, abbiamo assegnato innanzitutto un valore alla variabile. Dopodiché, abbiamo riportato l'operatore di addizione subito dopo quello di assegnazione, per poi mandare in output il valore.

La funzione `var_dump()` che presto imparerai a conoscere meglio serve a darti informazioni sia sul valore di una variabile che sul suo tipo.

Ecco il risultato del "dump".

```
int(4)
```

Fantastico! A quanto pare anche l'universo è salvo. Cosa abbiamo? Un quattro, pare. Ok, in effetti ha senso. Se ci pensi, `$a + $b` ritorna la somma di due variabili, non effettua un'assegnazione, mentre l'operatore `=` serve proprio a questo.

In poche parole, il valore in `$panda` viene incrementato di `1`, il valore a destra dell'operatore.

La parola "incrementato" non è scelta a caso. Quello che abbiamo davanti, infatti, è detto "operatore incrementale". Così come c'è l'incrementale, inoltre, abbiamo anche il "decrementale", che fa l'operazione opposta.

Lo scriviamo, però, in modo leggermente diverso.

```
1  <?php
2
3  // Assegno un valore.
4  $panda = 3;
5
6  // Decremento il valore.
7  $panda--;
8
9  // Dump del valore.
10 var_dump($panda);
```

Al posto di scrivere `$panda -= 1` ho usato la notazione veloce `--`, che si può usare SOLO se stai sottraendo `1` (oppure `++`, se stai aggiungendo `1`).

Il risultato? Eccolo.

```
1  int(2)
```

Al momento non ti preoccupare degli operatori `*` e `/`.

Invece, cosa succede se decido di mettere l'operatore `-` o `+` del `=`?

Proviamo.

```php
<?php

// Assegno un valore.
$panda = 3;

// Decremento di uno.
--$panda;

// Dump del valore.
var_dump($panda);
```

La risposta?

```
int(2)
```

Ah! Ma è la stessa cosa?

> Non capisco...

Beh, in realtà non è esattamente la stessa cosa. Una differenza c'è, ma l'esempio per spiegartelo è quello sbagliato.

Guarda questo di seguito. Assegnerò un valore e ne stamperò il valore *prima*, *durante* e *dopo* l'operazione di "decremento".

Pronto?

```php
<?php

// Imposto il valore.
$panda = 3;

// Dump PRIMA.
var_dump($panda);

// Dump DURANTE.
var_dump(--$panda);

// Dump DOPO.
var_dump($panda);
```

Eseguiamo il codice per scoprirne il risultato.

```
int(3)
int(2)
int(2)
```

Il primo valore è tre. Quello atteso, immagino. Poi, nel "durante" il valore rilevato è due. Bene. Infine, nel "dopo" il valore rimane due. Fin qui tutto ok.

A questo punto possiamo cambiare il posto di `--`. Così.

```
<?php

// Imposto il valore.
$panda = 3;

// Dump PRIMA.
var_dump($panda);

// Dump DURANTE.
var_dump($panda--);

// Dump DOPO.
var_dump($panda);
```

Guarda adesso il risultato:

```
int(3)
int(3)
int(2)
```

Visto? Allora c'è una differenza!

In poche parole, se specifichi l'operatore **prima** della variabile questa verrà decrementata (o incrementata) prima di eseguire l'istruzione sulla riga in cui ci si trova. Al contrario, se l'operatore viene specificato **dopo** il cambio di valore verrà eseguito solo dopo l'esecuzione delle istruzioni della riga corrente.

Riassumendo:

- **$value–** - Cambia il valore *DOPO* dell'esecuzione della riga attuale.
- **–$value** - Cambia il valore *PRIMA* dell'esecuzione della riga attuale.

In alcuni casi può essere abbastanza utile. Guarda qui:

```php
<?php

// Assegno un valore.
$panda = 3;

// Assign, quindi incremento.
$pandaFriend = $panda++;
```

Ci hai fatto caso? Ho praticamente risparmiato una linea di codice: si tratta fondamentalmente di una scorciatoia. Senza, avrei dovuto scrivere:

```php
<?php

// Assegno.
$panda = 3;

// Assegno il valore di $panda a $pandaFriend.
$pandaFriend = $panda;

// Incremento $panda.
$panda = $panda + 1;
```

Ci sono anche altri usi possibili per questo operatore. Li vedremo nel dettaglio, successivamente.