Introduzione

Era il 1998, avevo 12 anni e i miei genitori avevano appena acquistato il nostro primo vero PC. Non passò molto tempo e avevo imparato a modificare il codice di uno dei miei "sparatutto" preferiti: piccole cose, come fare in modo che il lanciamissili lanciasse cento missili al secondo anziché uno, poi fargli lanciare cento missili in tutte le direzioni… Fu così che iniziai a programmare. Il gioco era multiplayer e anche altri avevano imparato a modificarlo, così la battaglia si fece sempre più dura. Qualcuno mi sparò addosso cento missili tutti insieme. Avevo preparato uno script per creare una parete proprio davanti a me, in modo da bloccarli tutti. E andammo avanti così con attacchi sempre più potenti e difese sempre più efficaci. È così che ho appreso le basi della programmazione e ho scoperto che i veri limiti, in programmazione, sono l'immaginazione e la creatività. Nel frattempo, studiavo anche il linguaggio HTML e avevo anche un sito web nel quale condividevo le mie tecniche e i miei script di hacking dei giochi. No, quel sito web non c'è più.
Dal 2000, ho cominciato a studiare le basi di PHP/MySQL: gestivo un sito web per un gruppo di amici. Ho realizzato i miei primi rozzi script PHP per inviare articoli al sito web, per indire sondaggi e perfino per gestire i punteggi dei nostri tornei. A quel punto cominciai a scrivere applicazioni desktop in un terribile linguaggio di programmazione di nome Delphi, preparando strumenti che permettevano di eseguire il modding di vari giochi. Mi sono laureato nel 2007 con una tesi sull'Ingegneria del software, ho lavorato per varie società come sviluppatore PHP e ora sono tornato all'università, ma come docente, per diffondere la mia passione per la programmazione.
Imparare a programmare regala molte soddisfazioni. Potete osservare il vostro programma mentre prende vita mano a mano che lo costruite. Tuttavia, programmare può anche essere un'esperienza davvero frustrante. In questo libro, intendo usare la mia esperienza per offrirvi un percorso di apprendimento più lineare rispetto al mio e a quello della maggior parte degli sviluppatori. Fin da subito vi indicherò quale direzione prendere. Ma prima di tutto voglio fornirvi alcuni suggerimenti generali sulla programmazione e sul suo apprendimento, gli stessi che do ai miei studenti.

A chi è rivolto questo libro

Questo libro è rivolto ai web designer di livello intermedio e avanzato che desiderano compiere un salto di qualità nella programmazione lato-server. Ho immaginato di poter

dare per scontate le basi del linguaggio HTML e quindi me ne servirò senza dare troppe spiegazioni. Non è invece necessaria alcuna competenza nei linguaggi CSS (*Cascading Style Sheets*) o JavaScript, ma se conoscete JavaScript l'apprendimento di PHP sarà per voi un gioco da ragazzi, dato che questi linguaggi sono abbastanza simili.
Entro la fine del libro, conoscerete tutte le implicazioni legate alla costruzione di un moderno sito web in PHP, le basi del linguaggio PHP e le tecniche impiegate oggi da tutti gli sviluppatori.

La programmazione è cambiata molto

Per uno sviluppatore web alle prime armi, oggi, c'è davvero molto da sapere prima di poter mettere in piedi un vero sito web rispetto a quanto poteva accadere anche solo nel 2001. Un tempo era tutto molto più semplice. Per esempio, la sicurezza di un sito web non era poi così importante. A meno che non si trattasse del sito di una banca o di una società che accettava pagamenti con carta di credito, c'erano poche possibilità che qualcuno potesse prendere di mira proprio il vostro sito. Al giorno d'oggi, al contrario, ogni singolo sito web viene costantemente bersagliato da *bot* e script che cercano di sfruttare anche le più piccole "fessure" che potete aver lasciato inavvertitamente aperte.
Anche il modo in cui si programma in PHP è cambiato radicalmente, certamente in meglio. Ora è molto, davvero molto più facile riutilizzare nel proprio progetto parti di codice sviluppate da altri programmatori. L'unico svantaggio è che è necessario conoscere molto meglio i concetti legati alla programmazione prima di poter fare qualcosa di utile.
Per stare al passo con la concorrenza e con le esigenze dei progetti più impegnativi, anche PHP e MySQL hanno dovuto evolversi. Oggi PHP è un linguaggio molto più complesso e potente di quanto non fosse nel 2001, e lo stesso vale anche per MySQL. Imparare a usare PHP e MySQL, oggi, apre molte porte, che rimarrebbero chiuse a chi si fosse "fermato" al 2001.
Questa è la buona notizia. Quella cattiva è che, così come è più facile usare un coltello da burro rispetto a un coltellino svizzero (e con meno probabilità di farsi male), tutte queste nuove funzionalità hanno indiscutibilmente reso PHP e MySQL più difficili da imparare per i principianti.

10.000 ore per diventare esperti

La perentorietà di questa affermazione è discutibile ma l'idea di fondo è corretta. La programmazione è una competenza davvero difficile da padroneggiare. Non ci si può aspettare di diventare esperti dall'oggi al domani. Entro la fine di questo libro, avrete una certa competenza nell'uso di PHP, ma vi rimarrà sempre molto da imparare, indipendentemente dal livello che avrete raggiunto.
Detto questo, nel programmare un po' di conoscenza di base può fare molto. Sarete sorpresi di quanto sia possibile fare anche con pochi strumenti. Scoprirete che, dopo aver appreso le basi, potrete ottenere quasi tutto ciò che desiderate. Sarà davvero molto poco quello che non potrete ancora fare, anche conoscendo solo una parte dei concetti di programmazione. I concetti più avanzati servono soprattutto per rendere il codice più efficiente, più veloce e molto più facile da scrivere e costruire.

Resistete alla tentazione del… salto in avanti

Lo ripeto più e più volte ai miei studenti che saltano le lezioni: i concetti di programmazione dipendono l'uno dall'altro. Nella maggior parte dei casi, è necessario acquisire i concetti precedenti prima di poter passare a quelli successivi. Chi cercasse di procedere troppo velocemente, saltando qualche concetto di base, si sentirà inutilmente confuso e in difficoltà. Non sono molti i concetti di programmazione che "stanno in piedi da soli": se vi sentite bloccati in un punto, spesso è perché non avete compreso a fondo un concetto basilare precedente. Non abbiate paura di tornare indietro e ripassare ciò che pensavate di conoscere "a sufficienza". Di solito è molto più veloce ripassare, comprendere e procedere piuttosto che intestardirsi su un concetto avanzato, che risulta incomprensibile solo perché vi mancano le basi.

Il fallimento del Concorde

Alla fine degli anni Settanta, i governi britannico e francese continuarono a finanziare lo sviluppo dell'aereo di linea Concorde, anche se stava perdendo enormi quantità di denaro. Il loro ragionamento era semplice: avevano speso talmente tanto denaro sul progetto che se lo avessero abbandonato avrebbero perso l'intero investimento. Certo, alla fine hanno perso molto di più, proprio perché hanno continuato a pompare denaro in un progetto ormai morto. Se si fossero fermati prima, avrebbero risparmiato molti soldi. Questo è il proverbiale *Concorde fallacy*. Ci sono momenti in cui è meglio decidere di limitare i danni che continuare a investire su un progetto già fallito.

Arriverà un momento in cui passerete ore e ore su qualcosa e, semplicemente, continuerà a non funzionare. Se dovesse accadere, fate un passo indietro e provate a risolvere il problema in un modo diverso. Utilizzate gli strumenti alternativi che avete a disposizione. La soluzione potrebbe non essere elegante, ma una volta trovato il problema può sempre essere modificata. Non abbiate paura di buttare via tutto e ricominciare. Agli inizi, finirete per scrivere un sacco di codice, cercando di adattarlo a quello avete scritto in precedenza, fino a costruire, gradualmente, un mostro. E finirete per non capire davvero che cosa stia facendo il codice. Ogni modifica diverrà impraticabile e vi sentirete frustrati. Anche un minimo cambiamento richiederà un duro lavoro, poiché modificando qualcosa in un punto si disturberà qualcosa in qualche altro punto. Quando vi accadrà, non temete di ricominciare da capo. Ho perso il conto del numero di volte in cui ho avviato un progetto da zero, dopo averlo completato solo parzialmente. Di solito si arriva allo stesso punto entro un paio d'ore, ma con un codice molto più ordinato e semplice. Tuttavia, tenete sempre da parte quel codice come riferimento; non eliminatelo.

Tutti, all'inizio, programmano in modo terribile. Chiedete a qualsiasi programmatore di guardare il codice che ha scritto agli inizi e si arrabbierà, anche se si tratta di programmi scritti solo pochi mesi prima.

Non state imparando il linguaggio PHP

Sì, avete letto bene. Questo libro è incentrato interamente su PHP e MySQL, ma non cadete nella trappola di pensare di dover imparare il linguaggio PHP. Certamente, *state*

imparando il linguaggio PHP, ma per essere più precisi sto usando PHP per *insegnarvi a programmare*. Quando si impara a guidare, non si impara a guidare una Toyota. Si imparano le tecniche di guida, applicabili a qualsiasi auto, anche se alcuni controlli sono posizionati in punti diversi. I concetti esposti si applicano a quasi tutti i linguaggi. Certo, ci saranno alcune differenze, ma i concetti rimarranno gli stessi. Una volta che saprete programmare con competenza in un linguaggio, in pochi giorni potrete raggiungere un livello di programmazione non trascurabile anche in un altro linguaggio. Quindi non leggete questo libro pensando "Sto imparando a programmare in PHP", ma piuttosto "Sto imparando a programmare".
È più importante ricordare i concetti che la sintassi. La *sintassi* corretta potete sempre trovarla con facilità, ma capire i *concetti* sottostanti è più difficile. Il che mi porta al mio prossimo punto.

Saper collocare parentesi graffe e punti e virgola è facile

All'inizio metterete costantemente parentesi tonde, parentesi graffe, punti e virgola e… praticamente tutto il resto nel posto sbagliato. Vi dimenticherete di inserire un simbolo o una lettera e l'intero programma non funzionerà. Questo può essere incredibilmente frustrante, all'inizio. Ma una volta capito come fare, vi renderete conto che adottare la sintassi corretta è la parte più facile. È facile perché è rigoroso. O è giusto o è sbagliato. O funziona o non funziona. La parte difficile, in realtà, è scrivere la logica del programma, parcellizzando un problema fin nelle sue parti più piccole in modo da poterlo "spiegare" al computer. Il computer vi dirà subito se la sintassi è sbagliata, ma non avrà modo di dirvi se le istruzioni sono quelle necessarie per risolvere il problema.

Non otterrai nulla dalla pianificazione

> Pianificando e pianificando… non combinerai mai nulla.
> Karl Pilkington

Se vi è capitato di leggere qualche testo di programmazione, probabilmente avrete sentito che è necessario dedicare molto tempo alla progettazione del codice, che occorre pianificare attentamente la logica del programma e il suo funzionamento, prima di scrivere anche una singola riga di codice. Troverete libri e articoli che propongono le loro metodologie di sviluppo; la chiamano "ingegneria dei requisiti": diagrammi per rappresentare visivamente il codice e tanti suggerimenti su come pianificare per bene il codice prima di scriverlo. Ecco un esempio.

Ci sono tre passaggi fondamentali da eseguire quando devi scrivere un programma.

1. Definire l'output e i flussi dei dati.
2. Sviluppare la logica per arrivare a quell'output.
3. Scrivere il programma.

> Ricorda che la scrittura del programma è solo l'ultimo passo della scrittura di un programma. Non è un gioco di parole. La costruzione fisica della casa è solo l'ultima fase della costruzione della casa; una corretta pianificazione è fondamentale prima di iniziare a costruire qualsiasi edificio vero e proprio. Scoprirai che in realtà scrivere righe di codice è una delle parti più facili del processo di programmazione. Se il tuo progetto è ben concepito, il programma, praticamente, si scriverà da solo; digitarlo diventerà quasi un mero ripensamento dell'intero processo.

Ora dirò qualcosa che farà trasalire la maggior parte dei programmatori: ignorate *completamente* questi consigli e dedicatevi invece alla programmazione vera. Quando lo dico in aula, i miei studenti tirano un sospiro di sollievo. Sono lì per imparare a programmare e il modo migliore per farlo è iniziare a programmare. Il problema fondamentale di questo consiglio è che trascura un fatto abbastanza ovvio: per progettare un software, è necessario sapere quali strumenti sono disponibili e quali problemi risolvono. Altrimenti, qualsiasi progetto creerete sarà privo di significato, se non saprete su quali strumenti potete contare. Supponiamo che non abbiate la più pallida idea di come si costruisca una casa. Non sapete usare un martello o una sega, come si progetta un tetto, quanto devono essere profonde le fondamenta, come progettare le tubature, quali materiali impiegare per quali parti della casa e così via. Potete dedicare al progetto tutto il tempo che volete e pianificare le cose nel modo più accurato possibile, ma se non sapete quali attrezzi potete impiegare e quali sono i loro limiti, finirete per creare un progetto irrealizzabile con gli attrezzi e i materiali disponibili. Se non sapete di aver bisogno di fondamenta di sei metri per una casa di tre piani, non potrete progettare una casa di tre piani.
Allo stesso modo, non potete progettare un programma per computer se non sapete programmare. Per dimostrarlo, ecco una storia tratta da una conferenza "TED Talk" di Ernesto Sirolli intitolata *Volete aiutare qualcuno? State zitti e ascoltate*.

> Era un progetto in cui noi italiani abbiamo deciso di insegnare agli Zambiani come coltivare il cibo. Siamo arrivati là, con semi italiani, nel sud dello Zambia, in questa valle assolutamente magnifica che scendeva lungo il fiume Zambesi e abbiamo insegnato ai locali come coltivare pomodori italiani, zucchini e…
> […]
> Ed eravamo sbalorditi dal fatto che i locali, in una valle così fertile, non avessero nessun tipo di agricoltura. Ma invece di chiedere loro perché non coltivavano niente, abbiamo detto semplicemente: "Grazie a Dio siamo qui. Giusto in tempo per salvare lo Zambia dalla fame".
> E ovviamente tutto in Africa cresceva meravigliosamente. Avevamo questi magnifici pomodori. In Italia, un pomodoro diventa di questa dimensione [e Sirolli mima con le mani le dimensioni di un pomodoro], in Zambia di QUESTA dimensione [e il segno delle mani diventa più quello di un grosso pompelmo]. Non riuscivamo a crederci e dicevamo agli Zambiani: "Guardate come è facile l'agricoltura". Quando i pomodori divennero belli, maturi e rossi, in una notte duecento ippopotami vennero fuori dal fiume e mangiarono tutto. Dicemmo agli Zambiani: "Mio Dio, gli ippopotami!". E gli Zambiani risposero: "Per questo non abbiamo agricoltura qui".
>
> [Traduzione italiana di Anna Cristiana Minoli e Simone D'Urso.]

Ernesto e la sua squadra sapevano esattamente che cosa dovevano fare. Hanno pianificato attentamente tutto e sono riusciti a ottenere il risultato desiderato. Tuttavia, tutta quella pianificazione e progettazione è andata sprecata a causa di qualcosa di inimmaginabile.
I programmatori non incontrano molti ippopotami nel loro lavoro, ma troverete molti ostacoli che non sarete in grado di prevedere; inevitabilmente vi imbatterete in qualche problema. Tutto il tempo dedicato alla pianificazione se ne andrà sprecato per l'equivalente di 200 ippopotami, che arrivano e si mangiano tutto il codice. Occorre buttare via tutto e ricominciare.
In questo libro, parlerò dei tanti "ippopotami" che potrete incontrare, ma davvero vi incoraggio a provare il codice voi stessi. Imparate con la pratica. Iniziate a programmare. Quasi certamente il risultato non funzionerà la prima volta, ma nel frattempo avrete imparato qualcosa di più. Riprovate con un approccio differente e finalmente otterrete qualcosa di funzionante. Non c'è modo di progettare un programma finché non si è a conoscenza dei problemi che è possibile incontrare e dei limiti degli strumenti disponibili.

Ok, ma progettare non è del tutto negativo

Per evitare un'ondata di messaggi di odio da parte di altri programmatori, concluderò questa Introduzione dicendo che, per i programmatori professionisti, dedicare del tempo a progettare il codice prima di costruirlo è fondamentale. Tuttavia, i professionisti stanno scrivendo del codice sul quale loro e altri potrebbero dover lavorare per anni o decenni a venire. Il loro codice deve essere scritto in modo da essere estensibile e comprensibile anche dagli altri programmatori.
In questo libro, rifletteremo sulla struttura del codice e impareremo a scrivere codice che sia riutilizzabile ed estensibile. Ma al momento non siete qui per scrivere codice che verrà utilizzato in progetti reali e dovrà essere sottoposto a rigorosa manutenzione negli anni a venire. Siete qui per imparare. Allora andate e trovate tutti quegli ippopotami. Imparerete di più dagli errori che dal codice che funziona immediatamente.
Il tempo che dedicate a pianificare il codice dovrebbe essere proporzionale alla vostra capacità di programmare. Se siete agli inizi, purché abbiate una certa comprensione di ciò che intendete far fare al programma, iniziate a scrivere il codice fino a ottenere il risultato desiderato. Potrete finire in vicoli ciechi e dover tentare un altro approccio, ma senza la sensazione di *aver sbagliato*, solo perché una scelta va contro il progetto al quale avete dedicato ore e ore di lavoro. Ciò che ho appena detto per il Concorde vale anche qui.
Per i primi capitoli, lasciatevi andare. Eseguite il codice e verificate se funziona. Provate a risolvere alcuni dei problemi che ho predisposto, prima di saltare alle soluzioni. Imparerete di più scoprendo da soli le soluzioni che digitando, passivamente, il codice funzionante. Man mano che cresceranno le vostre conoscenze, avrete una più solida comprensione degli strumenti disponibili e del modo in cui è meglio risolvere i problemi. Una volta raggiunto quel livello, potrete iniziare a pianificare le cose in modo più dettagliato, prima di iniziare a scrivere il codice.

Scarica i file degli esempi

Trovate l'archivio del codice degli esempi utilizzati nel libro all'indirizzo `https://github.com/spbooks/phpmysql6`. Le istruzioni sull'uso del codice sono riportate nell'Appendice A. Quando il codice viene presentato nel testo, il nome del relativo file di esempio viene indicato nella testatina del listato, come segue:

Listato 0.1 Nome file.

```
.footer {
    background-color: #CCC;
    Border-top: 1px solid 333
}
```