Introduzione

Era il 1998, avevo dodici anni e i miei genitori avevano appena comprato il nostro primo PC. Non mi ci volle molto per capire come alterare il codice di uno dei miei sparatutto in prima persona preferiti: piccole cose, come fare in modo che il lanciarazzi spari cento razzi al secondo invece di uno e poi fargli sparare un centinaio di razzi in tutte le direzioni... e infine mandare in crash il gioco. Mi ero appassionato ed è da allora che ho cominciato a programmare.

Il gioco era multiplayer. Anche altri avevano scoperto come modificare il codice e la "corsa agli armamenti" ebbe una rapida escalation. Qualcuno mi lanciava contro un centinaio di razzi? Avevo uno script pronto che creava istantaneamente un muro proprio di fronte a me, per bloccarli tutti.

Il mio avversario generava una dozzina di mine antiuomo sotto e attorno a me? Io disattivavo la gravità e saltavo, allontanandomi dall'imminente esplosione. Tutti potevano volare. Ma siamo arrivati al punto in cui la cosa non era neanche più divertente. Entravi nel gioco e qualcuno aveva preparato uno script per teletrasportarsi dall'altra parte della mappa, ucciderti all'istante e costringerti a rigenerarti, poi ripeteva la cosa una dozzina di volte al secondo. Naturalmente bloccavano anche i tuoi comandi.

Abbiamo scoperto dei modi anche per bloccare tutto questo genere di cose, ma alla fine la situazione è entrata in stallo. Chiunque fosse riuscito a entrare per primo nel gioco poteva assumerne il pieno controllo, e per quanto fossero buoni i tuoi script, non c'era più niente che potessi fare. È stato divertente, finché è durato.

È così che ho imparato le basi della programmazione e anche il fatto che l'unico limite è la tua immaginazione e creatività. In quel periodo, avevo anche studiato, da solo, il codice HTML e avevo il mio sito web in cui condividevo alcune delle mie tecniche e script di hacking dei giochi. No, quel sito non è più attivo. Ed era terribile, pieno di pessima grammatica e animazioni scadenti, ma all'epoca si faceva così.

Nel 2000 avevo imparato da solo le basi della programmazione in PHP e MySQL e gestivo un sito web per il mio gruppo di compagni di gioco. Ho scritto alcuni rudimentali script PHP per la pubblicazione di notizie sul sito web, nonché sondaggi e persino uno script per gestire le classifiche e le partite dei nostri mini-tornei.

Successivamente, ho cominciato a scrivere applicazioni desktop in un terribile linguaggio chiamato Delphi, preparando strumenti che aiutavano chi voleva a modificare i vari giochi. Mi sono laureato nel 2007 con una tesi sull'ingegneria del software, ho lavorato per varie aziende come sviluppatore PHP, poi sono tornato al mondo accademico per

conseguire un dottorato di ricerca in ingegneria del software. Attualmente insegno all'Università di Northampton, nel Regno Unito.
Ora ho 34 anni e ho programmato per la maggior parte della mia vita. Lo trovo divertente, qualcosa che mi piace molto fare. Sto scrivendo questo libro per condividere le mie conoscenze con voi e aiutarvi a evitare alcune trappole in cui è facile cadere.
Imparare a programmare è molto divertente e gratificante. Vedi il tuo programma prendere vita mentre lo realizzi. Tuttavia, può anche essere un'esperienza incredibilmente frustrante. In questo libro, cercherò di usare la mia esperienza per offrirvi una guida più costruttiva di quella che abbiamo avuto io e molti sviluppatori. In tal modo posso guidarvi nella giusta direzione fin dall'inizio.
Prima di presentarvi il codice, vi darò alcuni consigli generali sulla programmazione e sull'imparare a programmare, che poi sono i consigli che do a tutti i miei studenti.

A chi è rivolto questo libro

Questo libro è rivolto ai web designer di livello intermedio e avanzato che desiderino compiere un salto nella programmazione lato server. Dovreste sentirvi a vostro agio con il codice HTML, poiché lo impiegherò senza dare molte spiegazioni. Non presumo, e non è richiesta alcuna conoscenza dei linguaggi CSS (*Cascading Style Sheets*) o JavaScript, ma se già programmate in JavaScript, l'apprendimento di PHP diverrà un gioco da ragazzi, poiché le basi di entrambi i linguaggi sono abbastanza simili.
Alla fine del libro, avrete una buona comprensione di tutto ciò che è legato alla creazione di un moderno sito web PHP, delle basi di PHP e delle tecniche utilizzate oggi dagli sviluppatori. Ma soprattutto avrete tutto il necessario per costruire il vostro primo sito web.

La programmazione è molto cambiata

Come sviluppatori alle prime armi, dovrete conoscere molte più cose prima di poter pubblicare un sito web, rispetto a quanto avveniva nel 2001.
Quando ho iniziato, le cose erano molto più semplici. Per esempio, la sicurezza del sito non era una considerazione poi così importante. Se non eri una banca o un'azienda che accettava pagamenti con carta di credito, c'erano davvero pochissime probabilità che qualcuno prendesse di mira proprio il tuo sito.
Al giorno d'oggi, al contrario, ogni singolo sito web è costantemente bombardato da bot e script, che cercano in modo sistematico di sfruttare anche le più piccole porte che potresti aver lasciato aperte.
Anche il modo in cui si realizzano gli script PHP è cambiato radicalmente, sicuramente in meglio. Ora è davvero molto più semplice scaricare e utilizzare nel proprio progetto il codice scritto da qualcun altro. Lo svantaggio è che c'è bisogno di conoscere molto meglio i concetti di programmazione, prima di poter fare qualcosa di utile.
Per tenere il 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 articolato e potente di quanto non fosse nel 2001, e MySQL è un database molto più complesso ed efficace. I linguaggi PHP e MySQL oggi aprono molte porte che sarebbero rimaste chiuse per gli esperti di PHP e MySQL del 2001.

Questa è la buona notizia. Quella cattiva è che, così come è più facile (e meno pericoloso) usare un coltello da burro che un coltellino svizzero, tutti questi straordinari e nuovi miglioramenti e funzionalità hanno inevitabilmente reso più difficile l'apprendimento di PHP e MySQL per i principianti.

Ci vogliono diecimila ore per diventare esperti

La correttezza di questa affermazione è discutibile, ma l'idea è corretta. La programmazione è una competenza ed è incredibilmente difficile da padroneggiare. Non aspettarvi di diventare esperti da un giorno all'altro. Alla fine del libro avrete una buona conoscenza della programmazione PHP, ma ricordate che ci sarà sempre qualcosa da imparare, indipendentemente dal livello che avrete raggiunto.
Detto questo, nella programmazione un po' di conoscenza può fare già molto. Sarete sorpresi di quanto potete fare anche solo con pochi strumenti.
Scoprirete che, dopo aver appreso le basi, potrete fare quasi tutto ciò che desiderate. Ci sarà molto poco che non potrete fare, anche conoscendo solo una frazione dei concetti di programmazione oggi noti. I concetti più avanzati sono volti a rendere più efficiente, più veloce e più facile da scrivere il codice e anche molto più semplice da comporre.

Resistete alla tentazione di saltare i passaggi

È quello che ribadisco sempre ai miei studenti che saltano le lezioni. I concetti di programmazione si sovrappongono l'uno con l'altro. In molti casi, occorre imparare determinati concetti prima di poter passare a quelli successivi. Se provate a procedere troppo velocemente, vi confonderete inutilmente e vi complicherete solo le cose.
Non sono molti i concetti di programmazione che operano in modo isolato rispetto agli altri, quindi se vi sentite bloccati, spesso il motivo è che non avete compreso appieno un concetto precedente. Quando rimanete bloccati, non abbiate paura di tornare sui vostri passi e darvi una rinfrescatina su ciò che pensate di sapere già: di solito è più veloce che cercare a fatica di procedere.

L'errore del Concorde

Alla fine degli anni Settanta, i governi britannico e francese continuarono a finanziare l'aereo Concorde anche se stava ciò faceva perdere ingenti somme di denaro. Il ragionamento era che avevano già speso così tanto per il progetto; se lo avessero abbandonato, avrebbero perso tutto ciò che avevano investito fino a quel momento. Ovviamente, alla fine hanno perso molto più denaro, perché hanno continuato a buttarci dentro soldi. Se si fossero fermati prima, avrebbero risparmiato molto, nel lungo periodo. Questo è un errore "da manuale": ci sono momenti in cui è meglio limitare i danni piuttosto che continuare a operare su un progetto fallito.
Verrà un momento in cui avrete dedicato ore e ore di lavoro a qualcosa che non funziona. Quando vi capiterà, fate un passo indietro e provate a risolvere il problema in un altro

modo. Usate gli strumenti alternativi che avete a disposizione. La soluzione potrebbe non essere poi così elegante, ma se funzionerà, potrete sempre modificarla.
Non abbiate paura di buttare via tutto e ricominciare. All'inizio scriverete molto codice, cercando di adattarlo a ciò che avete già scritto e costruendo gradualmente un vero e proprio "mostro". E finirete per non capire più che cosa sta facendo il codice, che diventerà inestricabile e vi farà sentire frustrati. Anche il minimo cambiamento richiederà un'enorme fatica, poiché molto probabilmente pregiudicherà il funzionamento di qualcos'altro.
Quando vi accadrà, non abbiate paura di ricominciare da capo. Ho perso il conto del numero di volte in cui ho ricominciato un progetto da zero dopo averlo quasi completato. Di solito, in un paio d'ore riuscirete ad arrivare allo stesso punto in cui vi siete bloccati, ma con un codice molto più ordinato e sicuramente una migliore comprensione del suo funzionamento.
Tuttavia, vi consiglio vivamente di tenere quel codice come punto di riferimento, invece di eliminarlo.
Tutti, all'inizio, scrivono codice terribile. Chiedete a un qualsiasi programmatore di esaminare del codice che ha scritto agli inizi e rabbrividirà, anche se ha iniziato solo pochi mesi prima.

Non impareremo a programmare in PHP

Sì, avete letto bene. Questo libro è incentrato interamente su PHP e MySQL, ma non cadete nella trappola di pensare che state *imparando a programmare in PHP*. Sì, certo, imparerete a programmare in PHP, ma la verità è che userò il linguaggio PHP per insegnarvi *come* si programma.
Quando fate le guide per la patente, non imparate a guidare una Fiat. Imparate i concetti della guida e potrete applicarli a qualsiasi altra auto, anche se alcuni dei controlli si troveranno in un posto differente.
I concetti che apprenderete sono applicabili a quasi ogni altro linguaggio che vorrete imparare in futuro. Certo, ci saranno alcune differenze, ma i concetti alla base rimarranno gli stessi.
Una volta che sarete in grado di programmare correttamente in un linguaggio, in pochi giorni potrete raggiungere un livello non banale di efficienza 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 afferrare i *concetti* che la *sintassi*. La sintassi corretta la potete sempre cercare da qualche parte, ma quello che è più importante è comprendere i concetti sottostanti. Il che mi porta al punto successivo...

Imparare a mettere le parentesi graffe e i punti e virgola è la parte facile

All'inizio, metterete costantemente le parentesi, le parentesi graffe, i punti e virgola, i punti e praticamente tutto il resto nel posto sbagliato. Vi dimenticherete di inserire un certo carattere, e l'intero programma non funzionerà.

Questo può essere incredibilmente frustrante, all'inizio. Ma una volta capito come funziona, vi renderete conto che saper usare correttamente la sintassi è la parte facile. È facile perché è rigorosa: o è giusta, oppure è sbagliata. O funziona oppure non funziona. La parte difficile è scrivere la *logica*, scomporre un problema in parti, in modo da poterlo spiegare al computer. Il computer vi dirà subito se la sintassi è sbagliata, ma non ha modo per dirvi se gli avete dato le istruzioni corrette per risolvere il problema in questione.

Non otterrete nulla dalla progettazione

> Non otterrete nulla dalla progettazione.
> – Karl Pilkington

Se avete già letto qualcosa sulla programmazione, probabilmente avrete sentito che occorre dedicare molto tempo alla progettazione del codice, che dovreste progettare attentamente la logica del programma e il suo funzionamento prima di scrivere anche una singola riga di codice. Troverete libri e articoli che insegnano varie metodologie di sviluppo, qualcosa chiamato "ingegneria dei requisiti", diagrammi per rappresentare visivamente il codice e ogni tipo di suggerimento su come progettare il codice prima di scriverlo. Per esempio, potete trovare questo consiglio in *Teach Yourself Beginning Programming in 24 Hours*:

> Ci sono tre passaggi fondamentali da seguire nello scrivere un programma:
>
> 1. definire l'output e i flussi di dati;
> 2. sviluppare la logica per ottenere quell'output;
> 3. scrivere il programma.
>
> Notate che scrivere il programma è solo l'ultimo passo nella scrittura del programma. La cosa non è così sciocca come sembra. Ricordate che l'edificazione fisica di una casa è solo l'ultima fase della sua costruzione; è fondamentale una corretta progettazione prima di poter iniziare a erigere qualsiasi edificio. Scoprirete che in realtà scrivere e digitare le righe del programma è una delle parti più facili del processo di programmazione. Se il progetto è ben realizzato, il programma si scrive praticamente da solo; digitarlo diventa solo quasi un ripasso dell'intero processo.

Ora sto per dire qualcosa che farà sussultare la maggior parte dei programmatori: ignorate del tutto quel consiglio e buttatevi nella scrittura del codice.
Quando lo dico a lezione, i miei studenti tirano un sospiro di sollievo. Sono lì per imparare a programmare, e il modo migliore per farlo è iniziare a scrivere.
Il problema fondamentale di questo consiglio è che trascura un fatto che in realtà è ovvio: per progettare il software, è necessario sapere quali strumenti avete a disposizione e quali problemi risolvono, altrimenti, qualsiasi progetto che vi verrà in mente non avrà senso. Ma seguiamo quel consiglio della citazione precedente: supponiamo che non sappiate nulla di come si edifica una casa. Non sapete usare sega e martello, non sapete quanto deve reggere una trave per sostenere il tetto, non avete alcuna idea di quanto debbano essere profonde le fondamenta, non sapete prevedere i passaggi delle tubature dell'acqua, non conoscete quali materiali sono adatti per quale parte della casa…

Potete dedicare tutto il tempo che volete a progettare le cose il più attentamente possibile, ma se non sapete come funzionano gli strumenti e quali sono i loro limiti, vi ritroverete con un progetto che non utilizza appieno gli strumenti o che, semplicemente, è inattuabile, dati gli strumenti e i materiali disponibili. Senza sapere che avete bisogno di fondamenta di sei metri per una casa di tre piani, non potete progettare una casa di tre piani.
Allo stesso modo, non potete progettare un programma se non sapete programmare!
Per dimostrare questo mio punto di vista, ecco una storia tratta da un discorso TED intitolato *Want to help someone? Shut up and listen*, di Ernesto Strolli (`https://www.ted.com/talks/ernesto_sirolli_want_to_help_someone_shut_up_and_listen`):

> Avevamo un progetto: noi italiani abbiamo deciso di insegnare agli zambiani come si coltiva il cibo. Così siamo arrivati lì, nel sud dello Zambia, con i nostri semi italiani, in una valle assolutamente magnifica che scende fino al fiume Zambesi. E siamo rimasti stupiti che la popolazione locale, in una valle così fertile, non avesse sviluppato alcuna agricoltura. Ma invece di chiedere loro come mai non stavano coltivando nulla, abbiamo semplicemente detto: "Grazie a Dio siamo qui. Giusto in tempo per salvare il popolo zambiano dalla fame". E, naturalmente, tutto in Africa è cresciuto magnificamente e abbiamo ottenuto bellissimi pomodori. In Zambia, i pomodori sono cresciuti ancora più grandi che in Italia. E dicevamo agli zambiani: "Guardate come è facile l'agricoltura!". Quando i pomodori erano belli, rossi e maturi, una notte, circa duecento ippopotami sono usciti dal fiume e se li sono mangiati tutti. E abbiamo detto agli zambiani: "Oddio! Gli ippopotami!". Al che gli zambiani: "Sì, ecco perché qui non coltiviamo nulla".

Il team di Ernesto *sapeva esattamente* quello che stava facendo. Ha pianificato attentamente tutto ed è riuscito a ottenere il risultato desiderato. Tuttavia, tutta quella pianificazione e progettazione è andata sprecata a causa di qualcosa che non avevano previsto.
I programmatori di norma non incontrano ippopotami, ma troverete molti ostacoli che non sarete in grado di prevedere. Ma, inevitabilmente, li incontrerete. Tutto il tempo che dedicate a progettare andrà sprecato quando l'equivalente di duecento ippopotami verrà e "mangerà" tutto il vostro codice. Dovrete buttare via il progetto e ricominciare da capo.
Nel corso di questo libro, vi avviserò dei vari ippopotami che potreste incontrare, ma vi consiglio di provare le cose voi stessi. Imparate facendo. Buttatevi. Scrivete del codice. Quasi certamente non funzionerà la prima volta, ma nel farlo avrete imparato qualcosa. Riprovate con un nuovo approccio e troverete qualcosa che funziona.
Non c'è modo di progettare un programma finché non si è consapevoli dei problemi che potreste incontrare e dei limiti degli strumenti disponibili.

Ok, ma progettare le cose non è "il male"

Per prevenire l'ondata di messaggi infuriati di altri programmatori, concluderò questa parte dell'Introduzione dicendo che, per *i professionisti programmatori*, dedicare del tempo alla progettazione del codice prima di crearlo è di *vitale importanza*. Tuttavia, i professionisti stanno scrivendo codice con il quale potrebbero trovarsi a lavorare per anni o decenni a venire. Il loro codice deve essere scritto in modo tale che sia facilmente estendibile e comprensibile anche ad altri.
In questo libro vi farò riflettere sulla struttura del vostro codice e su come scrivere codice riutilizzabile ed estendibile. Ma non siete qui per scrivere codice che verrà utilizzato in

grandi progetti e che dovrà essere sottoposto a manutenzione negli anni a venire. Siete qui per imparare. Andate e trovate tutti quegli ippopotami. Imparerete di più dagli errori che da un pezzo di codice che funziona subito.
Il tempo che dedicate a progettare il codice dovrebbe essere proporzionale alla vostra capacità di programmare. Se siete agli inizi, ma sapete bene quello che deve fare il vostro programma, iniziate a scrivere il codice finché non fa proprio quello che volete. Potrete rimanere bloccati e dover provare un altro approccio, ma non sentite di *sbagliare*, solo perché quello che fate è contrario a quel progetto sul quale avete lavorato per ore. Ciò che ho detto a proposito del Concorde vale anche qui.
Per i primi capitoli, almeno, tuffatevi nelle cose. Eseguite il codice, provate se funziona. Provate a risolvere alcuni dei problemi che vi sottopongo, prima di conoscere le soluzioni. Imparerete di più scoprendo voi stessi le soluzioni, che digitando alla cieca il codice che vi propongo.
Man mano che si amplieranno le vostre conoscenze, avrete una comprensione più solida di quali strumenti avete a disposizione e del modo in cui devono essere risolti i problemi. Una volta raggiunto quel livello, potrete iniziare a progettare le cose in modo più dettagliato, prima di iniziare a scrivere il vostro codice.

Dove trovare aiuto

PHP e MySQL sono "bersagli mobili", quindi è probabile che, nel momento in cui leggerete queste pagine, alcuni dettagli o alcune di queste tecnologie siano cambiati rispetto a quanto descritto in questo libro. Per fortuna, SitePoint ha una fiorente comunità di sviluppatori PHP pronti ad aiutarvi in caso di problemi, e manteniamo anche un elenco di errori individuati in questo libro, che potete consultare per gli ultimi aggiornamenti.

I forum di SitePoint

Nei forum di discussione dell'editore originale inglese, SitePoint, potete porre domande su tutto ciò che riguarda lo sviluppo web. Potete anche rispondere alle domande. Come funziona un sito di forum di discussione? Alcune persone chiedono, altre persone rispondono e la maggior parte delle persone fa un po' entrambe le cose. Condividere le vostre conoscenze avvantaggia gli altri e rafforza la comunità. Vi troverete molti web designer e sviluppatori divertenti ed esperti. È un buon modo per imparare cose nuove, avere risposte in fretta e divertirsi.
I forum SitePoint includono forum separati per PHP e MySQL:

- `https://www.sitepoint.com/community/c/php/31;`
- `https://www.sitepoint.com/community/c/databases/38`.

Scarica i file degli esempi

Man mano che procederete nella lettura di questo libro, noterete una serie di riferimenti all'*archivio del codice*. Si tratta di un repository GitHub che contiene ogni singola riga di codice sorgente di esempio presente in questo libro. Se andate di fretta (o volete rispar-

miare ai vostri polsi la sindrome del tunnel carpale), scaricate l'archivio (`https://github.com/spbooks/phpmysql7`). Selezionate l'esempio dal menu a discesa *Branch*, quindi scegliete *Clone or Download* e potrete scaricare un file `.zip` per tale esempio.
In alternativa, se avete familiarità con Git, potete clonare il repository.
Tutto il codice è scaricabile anche dal sito di Apogeo all'indirizzo `https://bit.ly/apo-sapm`.

I vostri commenti

Se non riuscite a trovare una risposta nei forum, potete segnalare e discutere i problemi nel repository GitHub del libro. Per segnalare problemi relativi al libro, potete scriverci a `books@sitepoint.com`. Sono particolarmente benvenuti i suggerimenti di miglioramenti e gli avvisi di eventuali errori che potreste trovare.

Cominciamo

Adesso che mi sono presentato, che vi ho dato alcuni suggerimenti generali e che vi ho mostrato dove trovare aiuto, è ora di iniziare! Partirete impostando l'ambiente di sviluppo e presto vi troverete a scrivere le vostre prime righe di codice.

Esempi di codice

Il codice in questo libro viene presentato utilizzando un carattere a larghezza fissa, in questo modo:

```
<h1>Un magnifico giorno d'estate</h1>
<p>È stata una giornata incantevole per una passeggiata nel parco.
Gli uccelli cinguettavano e i bambini erano tutti a scuola.</p>
```

Dove occorre riportare il codice preesistente, invece di ripeterlo tutto, ho usato dei puntini di sospensione:

```
function animate()
{
    ...
    new_variable = “Hello”;
}
```

Alcune righe di codice dovrebbero essere inserite in una sola riga, ma abbiamo dovuto mandarle a capo, indentate, a causa dei vincoli di impaginazione.

```
URL.open("https://www.sitepoint.com/responsive-web-design-real-user-testing/?responsive1");
```