前言

上一本書《*Soft Skills* 軟實力：軟體開發人員的生存手冊》（碁峰資訊出版，2017 年）彷彿才剛誕生不久，我自己都沒想到會這麼快又催生出這本新書。

好吧，我來猜猜，感覺上這兩本書之間相距的時間好像也沒這麼短。《*Soft Skills* 軟實力》英文版是 2014 年 12 月出版，而我從 2016 年夏天開始著手撰寫這本新書的英文版。然而，對我來說，好不容易寫完一本「鉅著」，才休息一年半的時間怎麼夠，因為——寫書是一項非常艱鉅的工作。

當然，如果你能完成這項艱鉅的任務，就能獲得一本印有「你的名字」的著作放在自己的書架上，可是催生一本書的過程絕對不是大家所想的那麼輕鬆愉快。

既然如此，你現在心裡可能會冒出一個問號，那這本書的作者幹嘛要再寫一本書？而且還在這麼短的時間內又出了一本新書？（至少以我的標準來看，我很快又出了一本書）

這絕對不是因為我缺錢！從經濟層面來看，一定有更多比寫書更好賺的方法，不如把時間花在那些方面還比較實際。而且，也絕對不是因為我很愛寫作。雖然我承認，有時我也蠻享受寫作過程中帶給我的樂趣，但在大部分的時間裡，絕對是痛苦多於快樂。

那麼，我為什麼要踏上另一段寫作旅程？這又不能讓我大賺一票，還要投入大量的個人時間，而且還伴隨某種程度的痛苦。主要原因是「我必須寫」。

在我遍覽過所有為軟體開發人員寫的書後，我發現沒有一本書能像字典一樣，從 A 到 Z 徹頭徹尾地納入軟體開發人員需要的所有知識，不只是如何進入業界，還有怎麼在職場上生存、讓自己進化，甚至是知道做哪些事才能讓自己成為一位成功的軟體開發人員。

我在 YouTube 上開設的頻道裡，影片下方的留言收到了數千個來自世界各地視聽者的問題，他們都是軟體開發人員，而且有男有女、有老有少、有資深的從業人員也有剛入行的新人，他們都在問各種跟軟體開發有關的問題，而且都圍繞著「軟實力」這個主題。

舉幾個我收到的問題，像是：

如何成為一位軟體開發人員？

如何學習技術能力？

怎麼談薪水？或者是如何在約聘和正職工作之間抉擇？

在職場上如何向上管理老闆？怎麼和同事、科技業女性相處以及處理職場歧視議題？女性如何在科技業生存？

想成為一位軟體開發人員，我真正需要知道的事有哪些，以及我該怎麼學習這些知識？

我應該上大學？參加程式研習營？還是自我學習？

如果我沒有任何經驗，怎麼找工作？

如何通過工作面試？

如何穿出個人魅力？

如何發展自己的職涯，進化到下一個階段？

事實上不只這些問題，我還能舉出更多例子。

談到這裡，我先說個壞消息，就是我還是找不到有哪個資源能讓我跟軟體開發人員說：你們只要擁有這項資源，它就能給你們所有這些重要問題的答案。那好消息是什麼？就是你現在手上捧著我的新書，而它就是那個資源。

所以啦，儘管我對於再寫一本新書的態度有所保留，特別是要我在這麼短的時間內完成（至少我自己是這麼認為），但我終究還是決定要寫這本書。這不只是因為我必須寫，當然我也不否認因為有點手癢才回來寫作，但最主要的原因還是因為我堅信，當你發現自己需要某項資源，此時，你能做的就是去找出它或是自己創造出這項資源。我找不到，所以我選擇創造。

希望你透過本書，和我一起加入這趟職涯發展旅程。

本書適合我嗎？

關心自身事業發展的你站在這裡，瀏覽到這本新書時，精彩的封面吸引了你的注意，讓你拿起（或點擊）書本翻閱。

翻著書頁的你自問：「這本書適合我嗎？」別擔心，就算我對你一無所知，我也能肯定地說：本書就是為這樣的你而寫。你會問我：你怎麼知道？畢竟，我連你有沒有機會翻到這本書都不知道。

好吧，我其實知道你會讀到這本書，因為如果你沒有，我的文字就不會像現在這樣，在這個非常時刻，神奇地化為聲音，出現在你腦海裡。我還知道和你有關的另一件事。既然你能讀到這裡，表示你還能忍受我的幽默，所以你和我一樣，都有敏銳的幽默感。好吧，不扯了，在你對本書失去興趣，將它放回書架上或是點擊瀏覽器的其他視窗頁之前，我該認真回答你的問題。

剛剛說的玩笑話都先放一邊。不管你在軟體開發業的職涯發展到哪個階段，本書都能帶給你某些想法。我快速地思考了一下可能對本書有興趣的讀者，並且歸納為以下三類，你可以從和自己最相關的範疇開始閱讀：

剛進入軟體開發業的新人或是有興趣學習軟體開發的讀者

如果你是在業界才剛起步、正努力學習軟體開發／程式設計的新人，或是已經擁有一點基礎但還沒找到第一份軟體開發工作的人，本書前兩部分的

章節內容對目前的你來說價值最高，因為我會在這些章節裡談：如何成為軟體開發人員以及如何贏得第一份工作。日後當你需要成為成功的軟體開發人員、在職場生存以及職涯再進化時，本書其餘部分的章節內容則有助於你填補任何知識上的落差。

此外，你還能在本書中找到更多相關的主題，這些都是你在其他軟體開發書籍裡看不到的內容（就我涉獵過書籍來看），能幫助你擺脫所有困惑，提供一些實際的做法，告訴你怎麼開始學習第一個程式語言，如何在前途未卜的大學、程式研習營和自主學習之間找到適合自己的方向。

中階開發人員

對這個階段的你來說，本書第三部分「軟體開發的基本概念」價值最高。這部分的章節內容能填補你可能存在的知識落差，幫助你主動管理自己的職涯，在職場上獲得成功。但不表示本書第一部分的章節內容就無法帶給你任何價值。就算你已經知道怎麼寫程式了，或許也會發現這部分的章節內容有助於你學習：如何進一步發展職涯、挑選新的技術能力、進修新的程式語言、應徵工作、打造吸睛履歷以及談到令你滿意的薪水。

此外，如果你有興趣讓自己的職涯再進化，事實上你也應該這麼做，那你會發現本書最後一個部分「職涯未來式」的章節內容也很有幫助。

資深的專業開發人員

我知道，我知道，你會說本書談的所有內容你都知道了。

這樣的你不需要一些所謂的「業界高手」來告訴你該怎麼當軟體開發人員，也不需要學習版本控制是什麼，或者聽聽應該去大學還是參加程式研習營這樣的見解。我懂，真的，請相信我。即便如此，我還是相信本書也適合這樣的你，因為：

首先，本書有一半的內容是針對開發人員的工作以及職涯再進化。就算你已經在職場一段時間了，或許也取得不錯的成績——對了，順便先恭喜你走到這一步，但你還是能從中受益，學習：如何以更好的方式應對同事和老闆、行銷自己的想法、培養領導力，甚至是加薪或升遷。

假使你還沒到達這個境界，或許已經發現自己的軟體開發職涯碰到無形的阻礙——「玻璃天花板」，在向上發展上遇到瓶頸。再這樣下去，你就只能說自己曾經待過這家公司，完成過幾個案子，也得到公司的員工紀念 *T-Shirt*，然後拍拍屁股走人，我也曾經陷入這樣的窘境。幸運的是，我突破了這道無形的玻璃天花板。我會在本書的部分章節裡談這一塊，告訴你：如何建立個人品牌、在研討會演講、發展兼職副業以及其他等等做法。

最後我要說，就算前面幾個部分的章節內容對你來說或許過於基礎，但你還是可以從中發現對你有價值的資訊，例如，學習技術、找到高薪工作、談到令人滿意的薪水，以及如何在約聘工作和正職工作之間抉擇。

再說，這個階段的你應該有在帶其他開發人員吧？將書中一些不錯的建議提供給他們，告訴他們如何開始著手，你不覺得也是很好的做法嗎？

所以，我要再強調一次，不管你是誰，都能在本書找到你的容身之處。

我到現在甚至都還在冒險。我可以大膽地說，就算你對軟體開發一點興趣都沒有，還是能從本書獲得成堆的價值。雖然這本書是專為軟體開發人員而寫，但內容實際上是談管理個人職涯，以及如何在職場獲得成功。

如果你已經讀到這，表示本書真的、真的很適合你，因為，很顯然地你喜歡我，而你知道的，我也喜歡你。

第 1 章

本書的使用方法

毫無疑問地，你一定注意到了，本書的篇幅很長。我算了一下，少說有二十萬二千餘字，真的很多。所以，我認為應該佔用一點篇幅，先說明本書**為何會有這麼多字數的原因**，以及**你如何從本書中獲得最大的效益**。

本書是根據開發人員職涯的各個階段來規劃章節內容，不管你是剛入行的新人、已經累積一些經驗或者是經驗豐富的專業人員，都可以發現不同部分的章節內容或多或少都能用在自己身上，隨著時間發展還能回頭重新來看某些部分或章節的內容。

本書的目的

你或許正在疑惑，**我為什麼決定要寫這本特別的書**（雖然我在前言裡已經簡短提過，但我想在這裡再次說明）。

在我的部落格（*https://simpleprogrammer.com/cg1-blog*）和 YouTube 頻道（*https://simpleprogrammer.com/cg1-youtube*），讀者和視聽者最常問的問題就是，如何成為一位軟體開發人員，以及如何擁有成功的職涯。

我尚未找到這樣一份完整的手冊，不僅能指導剛入行的新人又能引領資深開發人員，告訴他們如何在職涯裡如魚得水，還有處理在業界工作時一定會遇到的一些議題。我在前一本書《*Soft Skills* **軟實力**》裡曾經淺談過其中的某些主題，但我認為有相當多的人想更深入了解這些主題。

《*Soft Skills* **軟實力**》的焦點是放在軟體開發人員的整體生活，個人職涯只佔其中一小部分的內容，所以，**我撰寫這本新書專門談軟體開發人員的職涯**。

本書內容為獨立設計，所以，你不需要先讀過《*Soft Skills* **軟實力**》或其他書籍，只要閱讀本書就能從中獲得最大效益，甚至不需要擁有任何軟體開發的經驗。

本書的目標

首先，我想幫助準備進入軟體開發業的新人，協助你們學習所有需要知道的重要知識，幫助你們起步，跳入這個有時棘手又複雜的業界。我想提供軟體開發業的新人**一項資源，這項資源會納入業界的所有重要面向**，告訴你們在業界起步需要的知識，展示最好的做法給你們看，讓你們贏得第一份工作。在我的認知裡，軟體開發業新人進入業界時，都會面臨這些**最重要的掙扎**。

接著，我想幫助已經在業界工作的軟體開發人員，協助你們彌補知識上的落差。這些知識可能是你們目前缺少的部分，至少是跟職涯有關，並且提供一些指引，告訴你們身為軟體開發人員如何在業界生存。我還會告訴你們如何解決一些問題，這些問題跟如何在某些情況下達成平衡有關，例如，平衡生活與工作、如何進行團隊合作、開發人員如何行銷自己的想法、加薪和升遷，以及如何培養領導力和處理職場歧視。

最後，我想幫助處於所有職涯階段的軟體開發人員，協助你們的職涯再進化，**通往下一個階段**。我會談如何在軟體開發業界建立名聲、選擇不同的職涯發展路徑，以及應該閱讀的書單，還會納入兼職副業、研討會和其他等等主題，**幫助你們更上一層樓，成為優秀的開發人員**。

本書所有內容還是要歸類為軟實力，主要著墨於你理論上該知道的事和要會的技能，而非教你實務上的程式技巧。**在現今的社群和產業裡，我相信仍舊大幅缺少這些智慧**，而我堅信，相較於學習特定程式語言或架構，長遠來看，這些智慧更有價值。

本書內容分為五大部分，每個部分包含數個短篇幅的章節，和《*Soft Skills* 軟實力》的編排方式一樣。

- 起步
- 上工
- 軟體開發的基本概念
- 職場進行式
- 職涯未來式

本書的終極目標是，不管你身在軟體開發職涯的哪個階段，都能從本書獲得某些知識，幫助你前往開發人員職涯的下一個階段。

如何使用本書？

讀者該如何使用一本書，這問題似乎再明顯不過了。若要使用一本書，例如本書，最實際的方法當然就是拿起書，然後閱讀（如果你是買紙本書而且書本夠厚，還可以找到其他使用方法，例如，把書放在桌上，墊高你的電腦螢幕）。**你當然可以一頁一頁從頭讀到尾**，我想多數人閱讀本書時都會選擇這個方式，不過，**你也可以跳著讀每個部分或是每個章節的內容**。

我先假設你是準備發展軟體開發職涯的新人，甚至還沒真正地開始學習程式。在這個情況下，從「起步」開始看起，你能獲得最大的效益，這部分的內容不僅和你最為相關，碰巧又是第一部分。

但是，假設你已經是軟體開發人員，寫了好幾年的程式。你或許會想直接跳到「職場進行式」或「軟體開發的基本概念」，以填補你目前在任何知識上的落差。或者你只對職涯發展有興趣，可能立刻就有興趣閱讀。此時，你或許會想直接跳到「職涯未來式」這部分，以採取最明智的行動。

本書中的**每個章節都是獨立的**。所以，你可以從目錄裡挑選哪個章節適用於你，能回答你現在或未來所面對的問題。本書會採這樣的方式設計，是因為我知道在軟體開發人員職涯發展的過程之中，你會遇到哪些情況，關心哪些面向。

當你剛入行時，你會想知道怎麼起步；如果你正努力學習某個新的程式語言或某項新技術，或許會想要類似的建議。你或許現在不需要找工作、談薪水，也可能不需要應付不友善的同事或老闆，但是這些章節未來可能會和你所處的情況息息相關。

每次我想回頭閱讀某本書的某個部分卻想不起來這些內容在哪裡，因為它們被埋在某些章節裡，而我只記得自己讀過哪幾章，這總是讓我感到沮喪。所以，**我嘗試讓本書不僅能直接閱讀內容，也能作為你在軟體開發職涯發展過程中的一本參考手冊**。

反覆行動

下一章開始是本書真正的內容，在那之前，關於如何使用本書，我想再談最後一點。

首先，**對於本書所寫的建議，如果你不採取任何行動，本書對你來說一無是處**。閱讀本身是一件好事，你甚至可能還打從心裡同意作者的看法，可是，如果你不將書裡所學應用在生活之中，將無法發揮太大的作用。與其加重你的負擔，在「每個章節的最後出作業（這次沒有任何作業）」告訴你應該做什麼，或告訴你要記大量的筆記，或確定你應用了每天學到的一件事，我會給你更簡單的解決方案，這也是我自己親身實踐的方法。

反覆

如果你真的想改變自己的行為，並且在生活裡採用新原則和最佳實踐方法，最好的做法之一就是，**讓自己圍繞在這些你想融入生活的想法和觀念裡，讓大腦沉浸其中**。最佳的做法之一是利用「反覆」。讓自己以最沒壓力的方式來吸收和應用資訊。我一直以來都是採用這樣的做法。

我已經讀過某幾本書十幾次了，因為這些書對我的職涯和人生相當有價值，我真的想將這些書裡的觀念和信念內化於自身。所以，**我強烈建議你要一而再，再而三地閱讀本書中和你最相關的部分**，甚至是**在行事曆上提醒自己**，每年或是每隔一段對你最有效益的時間，都要重新再看一次這本書。

行動

我打算分享給你的所有想法和策略，除非你開始採取實際的步驟，並且付諸行動，否則這些想法和策略無法對你或你的職涯有好的影響。

為了簡化做法，我將蒐集的資源整合在一起，稱為「軟體開發人員職涯發展成功手冊數位套件（The Complete Software Developer's Career Guide Digital Toolkit，*https://simpleprogrammer.com/career-guide-toolkit*）」。這項工具套件裡包含一個按部就班的流程，能幫助你更快找到軟體開發人員的工作（即使你沒有任何經驗）；一套速成課程，能幫助你準備軟體開發人員的面試；一項入門指導手冊，能指引你上班時該怎麼穿，以獲得老闆和同事更多的尊重；一份「偵錯檢查表」，能幫助你獵殺討人厭的臭蟲……。

我保守估計這個工具套件裡的資源至少價值 175 美元，約五千元台幣。不過，現在只要購買本書就附贈這些資源，不會收取額外的費用。你可以從以下網址下載這個工具套件（*https://simpleprogrammer.com/career-guide-toolkit*）。

隨著你的職涯和人生變化與成長，我誠摯希望你能持續從本書獲得價值。來吧，讓我們正式起步……。

SECTION 1

起步

「如果你心裡已經有個夢想，當然可以窮盡一生為這個夢想學習、規劃與做好準備，但你真正應該做的是開始實踐。」

— Dropbox 創辦人暨執行長 Drew Houston

至今為止我收過來自四面八方的各種問題，在軟體開發方面，最常見的問題就是：我想進入軟體開發業界，該如何起步。

人生之中要真正動手去做某件事，並且達成我們心中的想望，最大的障礙似乎總是該如何起步。不管你是想開始規律的健身計畫、馬拉松訓練、創業還是寫書——更具體一點，就本書內容來說是想開始學程式設計，最困難的部分總是起步。

人們很容易陷入一種糟糕的情況——浪費掉無數的時間討論該做什麼。看別人怎麼做，總是比自己真正動手做來得輕鬆。一直想接下來要走哪一步，無止盡地討論該怎麼走，往往比實際邁出這一步要容易得多。

起步的秘訣就是你只要一步一步來就好。

你只要鼓起最大的勇氣和決心，對自己說：「我已經做了充分的討論，我對自己的想法有足夠的信心，或許不是最棒的計畫，但我也做好規劃了，無論如何，我都要去實踐。」而且，一旦你邁出第一步，就會在這條路上繼續走下去。不知

不覺，等你發現並且回頭一看，過去你走過的數千個步伐，已經帶你從山腳來到山頂。不過，在你開始動手之前，你還需要一個計畫。

許多想從事軟體開發工作的人根本就拒絕邁出第一步，還有一些人自以為很厲害，不蒐集資訊也不做規劃就一股腦地跳下去做，連自己未來的方向和目的地在哪裡都不知道。因此，本書 Section 1 的章節內容會告訴你，想成為軟體開發人員，剛開始要了解哪些基礎知識。我們會談如何擬定一套真正的計畫來幫助你成為軟體開發人員；想在程式界生存，你需要哪些技術能力以及你如何培養這些技術能力。

Section 1 的章節內容還會包含：該學哪種入門程式語言，以及最佳的學習方法——不論你想採自主學習、參加程式研習營或是走傳統路線去念大學，本書都會分析其優缺點。在最後一部分的內容裡，我的目標是讓你有足夠的知識起步，制定真正可以實踐的計畫和時程表。

如果你是現役的軟體開發人員，仍舊可以在這部分的章節裡找到對你有幫助的內容：像是補強現有知識、對未來職涯做更好的規劃，或者是決定如何在軟體開發方面持續進修的方法（你或許也會發現這些資訊能幫上你認識的某個人，這個人正苦苦掙扎想進入業界，而你正好可以提供一些指引）。

不管你是哪種情況，我都推薦你下載隨書附贈的「軟體開發人員技能評估工具」（Software Developer Skills Assessment，*https://simpleprogrammer.com/career-guide-toolkit*）。藉由這項工具，你可以快速找出自己需要補強的技術能力，並且成為更有自信的開發人員。

雖然我可以給你進入軟體開發業界的所有建議和資訊，告訴你該走哪條路，可是，除非你能自信地踏出第一步，否則一切都不會發生。就像我很喜歡說的一句話：你必須「相信過程」（trust the process），而我就是那個過程。那麼，就讓我們正式起步吧……

第 2 章

入門

我剛踏進軟體開發這一行時，根本不知道自己要做什麼。那時我覺得非常沮喪。**一切事情似乎都沒有意義**，我不認為自己能「獲得成功」。我會跟你提起這件往事的原因是，如果你正拿起這本書翻閱，或許會跟當年的我一樣，有相同的感受。別擔心，**這很正常**。**事實上，再自然不過了**。

有一點我要先說清楚：**不是只有天才**或是**智力高於平均水準的人**，才能成為軟體開發人員。如果你剛踏進軟體開發業界時，沒有感到不知所措（*https://simpleprogrammer.com/cg2-overwhelmed*），覺得像是腳踝上綁著重重的鉛塊，往游泳池深處跳進去，這樣的你要不是做錯了什麼，不然就是你不是人類，或許兩者都有可能。不管怎樣，剛踏進這個領域時，你的預期感受應該是艱難和困惑，但我向你保證，事情並不是永遠都會朝這個方向發展。

起點

我還記得當年我第一次自學程式時，手上並沒有現在的這些資源可以參考。事實上，**我那時手上沒有任何資源**。

我下載了當時很熱門的遊戲 MUD 的原始程式碼（Multi-User Dungeon 簡稱 MUD，指多使用者迷宮世界，這是一款沒有圖形介面的遊戲，你可以將它想像成文字版的**魔獸世界**。沒錯，那是個還是「以數據機撥接連線到 BBS 系統」

的年代）。我打開程式碼，根本不知道自己在看什麼，我只知道我想創造自己的 MUD 遊戲，新增自己想要的功能，而達成這一切的關鍵就埋在這堆怪異神秘字串的某處。

剛開始我手忙腳亂。先是將幾個變數修改成不同的變數值，然後找到了一些程式碼，似乎能控制給對手致命一擊的機率。於是，我修改了程式碼然後重新編譯一個新版本的 MUD，看看會發生什麼事。有時候會產生我想要的結果，**有時甚至連程式碼都無法編譯**，但我看到了哪些程式寫法有效、哪些無效，從中學會了一些知識。雖然我還是不知道自己在做什麼，但是在一週左右的時間內，我修改了程式碼並且設法創造出一個新版本的 MUD，裡面確實有一些我自創的「功能」。儘管離專業的程式設計師還有好長一段路要走，但這是一個開始——我們每個人都需要一個起點。

我想告訴你這個故事的原因是，**我認為比起挑一本教科書來看、去上大學或程式研習營等等其他更多的學習……這樣的方式才能真正地開始學習程式設計**。你必須動手修改程式，看看哪些結果有效，哪些不能用（我相信這是最佳的學習方式，請參見我前一本著作《*Soft Skills* 軟實力：軟體開發人員的生存手冊》裡 Section 3「學習」的章節內容，*http://simpleprogrammer.com/cg2-softskills*）

然而，**學習程式設計和了解如何踏入軟體開發業界是兩件非常不同的事**。沒錯，你需要學習怎麼寫程式，但軟體開發包含的內容更廣，不只有寫程式而已，本章內容就是要談這些「更廣」的內容。

學習專業知識

首先，你必須了解一些跟軟體開發有關的知識。可能比你想得還要簡單，也可能比你想得還要困難。

第一部分的所有章節內容的主旨是致力於「關於軟體開發，你需要了解哪些知識」，但我想先快速給你一個概念。**軟體開發不只是程式設計**。雖然其中一大部分是程式設計，但只知道如何寫程式，無法讓你在業界走得長遠，特別是如果你想在這個業界出人頭地。

多數軟體開發專案背後的想法是**將手動流程自動化**，或是創造嶄新的自動化方法來做某件很難手動做到的事。想想我現在正在用的文書處理軟體吧，我恰巧用了 Google Docs 輸入本章的內容。如果沒有 Google Docs 或其他的文書處理程式，我只能靠打字機輸入文件或是手寫。如果想將文件內容加上格式再印出來，我就必須手動排版要印出來的字母。要是想修正文件裡的錯誤，特別是修改拼字錯誤時，我就需要放一瓶立可白在手邊了（或許還需要一瓶威士忌）。

現在不只 Google Docs 能讓我做到這一切，還有一大堆硬體和軟體程式能讓我將一本書的手動打字或手寫流程自動化。我想你抓到重點了，我要強調的關鍵概念是，既然你踏上了程式人員的修業旅程，應該盡早開始學習。**在你能自動化處理某件事之前，必須先了解手動處理的方法**。

了解問題

太多有抱負、有經驗的軟體開發人員在完全不了解軟體應該要做什麼之前，就一股腦地投入軟體開發的工作，只想直接跳到寫程式的步驟（在我舉的 MUD 例子裡，從實作中學習程式是很好的做法，但不適合用於開發商業生產用的軟體）。既然你正在看這本書，顯然，**你比我想得還要聰明**。

軟體開發流程的第一步，永遠都是從了解你要解決的問題開始。不然你怎麼知道你要自動化什麼？不同的軟體開發方法論有各自不同的做法，但這不是我們現在要談的重點。現在的關鍵在於你寫任何程式碼之前，都必須以某些方法先蒐集某種需求，並且瞭解你要解決的問題。簡單一點的作法像是和潛在顧客聊聊，討論軟體需要開發哪些功能以及這些功能應該如何運作；正式一點的作法像是建立完整的規格文件。

設計

全盤了解需求之後，對於如何寫程式來解決這個問題，你可以提出一些設計。再次強調，這些動作都要在你寫程式之前進行。你可以將這一步想成是程式碼的架構藍圖。再次聲明，不同的軟體開發方法論在處理這一步上有不同的做法，但重點是**你跳進去開始寫程式之前，要有某種程度的設計**。

這個概念適用於大小規模的專案。一些學習敏捷軟體開發方法（我們會在後續章節談到）的開發人員認為他們不需要做任何設計，就能立刻開始寫程式。雖然敏捷開發的焦點是減少專案的前期設計工作，但**設計仍然有其必要**。你不能隨便拿 2x4 英吋的木條釘在一起，就說你要蓋房子。

撰寫程式碼

提出軟體設計上的一些想法後，就該寫一些測試程序來測試你定義的軟體功能（這個程序也稱為測試驅動開發，簡稱 TDD，後續章節會針對 TDD 有更深入的討論），或者是開始撰寫軟體本身的程式碼。**撰寫程式本身就是一門學科領域**，所以我們不會在此深入討論，但我想推薦兩本很棒的好書給你，都是針對如何寫出好品質的程式碼，你一定要找來看看。

首先，我要推薦 Steve McConnell 所著的《*CODE COMPLETE*：軟體開發實務指南》（Code Complete，*https://simpleprogrammer.com/cg2-codecomplete*）。這是一本所有軟體開發人員都應該閱讀的經典之作。第二本好書是 Robert Martin 所著的《無瑕的程式碼》（Clean Code），本書能幫助你提升程式碼的品質。這些好書有助於你學習如何撰寫程式碼結構，以及如何寫出更容易理解和維護的程式碼。**這兩本書都對我的程式技巧有深厚的影響**，特別是對程式清晰度和設計方面。

測試與部署

程式碼寫好之後，我們就可以發佈出去了，對嗎？

錯。**現在我們要來到測試程式碼的流程**。再說一次，不同的方法論對此會有不同的處理方法，不過，一般來說，程式提供給最終使用者之前都必須經過某種測試。例如，在傳統的瀑布開發專案裡，一直到專案非常後期才會進行測試，但在「敏捷」專案裡，每次迭代都會進行測試，一次迭代通常會持續兩週左右的時間。

程式碼測試完畢後，就可以準備部署了，部署本身是一項完整的流程。部署是指將已經完成的軟體安裝在伺服器上、上架到應用程式商店或者是以其他管道讓使用者接觸到軟體的完整流程（而且這套流程相當複雜）。本章不會深入部署流程的細節，後續我們會以一整章的篇幅來探討這個主題。

軟體部署過程中，程式碼可能——嗯，應該是一定要**簽入原始程式碼的資源庫裡**。這個資源庫負責儲存各個時期不同版本的程式碼和修改內容。多數處理大量資料的複雜應用程式，**可能還會導入某種類型的資料庫**。資料庫通常是用來儲存應用程式的使用者資料或設定資訊，可能需要跟著原始程式碼一起更新。許多軟體開發團隊還會利用某些類型的持續整合。當開發人員「簽入」部分程式碼時自動建置程式碼。

軟體開發不只有寫程式而已

最後，不要忘了，我們還要除錯。身為開發人員，我們有大量的工作時間是花在找出自己寫的或他人寫的程式碼為什麼不能運作的原因。**現在你知道了，除了寫程式之外，軟體開發還有很多事要做**。

在你找到一份真正的軟體開發工作之前，你需要知道這一切該怎麼運作。希望最好是能在這幾個專業能力上擁有一些經驗和技能。但是，別怕。**本書的目的**

就是要幫你做好一切的準備，不然，至少也要將你導向正確的方向。在你踏上旅程之前，你需要將背包裝滿你需要的一切用品，而我至少會告訴你需要打包哪些東西。

提前計畫

喔，John。我現在知道了，軟體開發這檔事不只是寫寫程式而已，我還要花大量的時間幫程式除錯，可是，你還是沒告訴我該如何在這個業界起步。這一切究竟是怎麼一回事？

啊，對喔。我知道你想問的重點，但是你猜猜看？先給你一個好消息：

你其實已經起步了，恭喜你。

你挑了這樣的一本書，努力想了解軟體開發的領域裡不只是寫程式而已，從那一刻起，你就已經**比以往多數軟體開發人員有一個更好的起跑點**。好吧，好吧，我知道這聽起來有點搞笑，但這也是實話。當你有一天跟我一樣，變成一個愛發牢騷的老軟體人時，你會說一樣的事。

現在，我們要從更務實的一面來看……你需要一個計劃。沒錯，一個計劃。你需要一個真正可以實踐的計畫，告訴你如何從對軟體開發一無所知或只懂皮毛的人，成長為羽翼豐滿的軟體開發人員。想到達這個境界，你有很多條路可走，我會在後續的章節裡介紹這些你能選擇的道路。然而，重點不是你選了哪一條路，是**你選了一條路之後還能堅持走在這條路上**。

規劃

接著，我們要來討論你的計劃應該包含哪些內容。

首先，你要**誠實地評估**自己現在的能力，以及未來需要學習哪些內容。

你曾經有過任何程式設計方面的經驗嗎？

你知道哪些程式語言？

你曾經建置過應用程式嗎？或是自己完全從頭開始？

你知道我先前提過的那些技能嗎？

有沒有哪個能力是你現在已經具備的？

你了解資料庫、版本控制、測試驅動開發、測試、除錯或軟體開發方法論嗎？

此外，你還要問問自己，**你想從事哪種類型的軟體開發工作**。

當然，每個人都想成為遊戲開發人員，但有可能實現嗎？你想從哪裡開始著手？走在這條漫長又寂寞的開發之路上，你願意長時間投入工作並且對抗所有的競爭情況嗎？**許多人從一開始就完全沒有想清楚人生的方向要朝哪裡前進**。請花點時間回答我剛提出的這些問題，你就能提出一個不錯的計畫，幫助自己起步。

請不要誤會，本書會盡力幫助你，但我也**只能帶你走到這裡**。我能提供一切你需要的資訊，告訴你如何成為一名良好、甚至是優秀的軟體開發人員，但是你必須**組織這些資訊，為自己量身訂做一套行動計畫**，然後依循這套計畫前進。

制定計畫

等你都想清楚這些問題了，就該來制定一套務實的計畫。

制定計畫時，最好的做法是**從你想達成的目標往回推**。你應該提出**一個具體的目標，清楚表示你想成為哪種軟體開發人員**，而不是模糊地說，「我要學程式設計」或「我想成為軟體開發人員。」之後我會在「軟體開發的基本概念」的章節內容裡，介紹各種不同的軟體開發角色或工作，或許可以參考看看；不過，你也可以自己先做一些調查，以決定哪種職務最適合你。**越具體越好**，這樣你才

能確實了解自己需要學什麼、如何打造履歷和作品集、需要進入什麼學校或課程學習，甚至是了解自己要應徵哪些工作。

我知道要做出決定和承諾是一件不容易的事，但這一步真的非常非常重要！如果你能越具體地說出你想成為哪種軟體開發人員，「**一切事情」都會變得更加容易**。你會清楚地知道自己需要學什麼，以及將來的每一步你需要做什麼。

誰想成為「運動員」?

這麼想吧：假使你說你想成為一名「運動員」。

天啊，你知道這範圍有多廣嗎？你要如何訓練自己成為一名「運動員」？你可能應該練舉重和跑步，但也可能應該練習游泳，拿網球拍來打個球或許也不錯。最好是把所有運動都練一練，越多越好，不管你最後要加入哪個運動團隊，你都能為所有運動做好準備。**你看，這聽起來是不是很可笑？**事實上，當某人說希望成為「軟體開發人員」時就是如此，聽起來不就跟這個例子一樣荒謬可笑。所以，**你要先挑出一項運動**。一旦你知道自己要從事哪種運動，你才知道要如何針對這項運動進行訓練。

相信我，這樣會讓你的生活更輕鬆。**從目標開始往回推**，你才能決定為了達成這個目標，你需要知道什麼和做什麼。完成這一步之後，就能來制訂計劃了。

計畫內容的開頭應該是你需要學習哪些項目。找出這些項目的學習順序和學習方法非常重要。再來是，針對應徵工作和獲得第一份工作，找出你應該做哪些準備工作？最後是針對找工作這件事，制定實際的行動計劃。你要去哪裡看工作？該怎麼求職？**打算應徵哪種工作？**我或許還會再加上一份計劃——獲得第一份工作之後，你要如何持續個人發展和教育訓練。

這些聽起來有點沉重，我懂，但別擔心。我寫這本書的理由就是，**讓你能更輕鬆面對這一切**。在接下來的幾個章節內容裡，我會幫助你找出自己需要了解哪些知識以及如何獲得這些知識；後續其他部分的章節內容會再詳細說明如何求

職。現在，你可以開始想想**你的計畫全貌會是什麼樣子**，並且努力找出**你想成為哪種開發人員**。

嗨，John

但是，我到底想當哪種開發人員？

這是一個好問題。如果你才剛起步，可能除了遊戲開發人員之外，你根本不知道自己有哪些職涯道路可以選擇。幸運的是，雖然你可能需要花點時間調查，但這不是太難理解的事。我會在本書後續的章節裡，討論一些軟體開發人員的類型。大部分的內容都在「軟體開發的基本概念」這個部分的章節裡，但你也可以自己做一些調查。

問問你認識的軟體開發人員，了解他們是從事哪種類型的軟體開發工作，或是他們本身是哪種開發人員。想想看你對開發哪種類型的事物有興趣，並且調查一下相關的技術和程式語言。

軟體開發人員可以關注的各種技術和職業分支非常廣泛。你想寫網頁應用程式？還是行動應用程式？你想寫程式讓冰箱能適當地控制溫度嗎？或許你想寫程式將太空人送進太空？仔細思考，然後調查。只要你能問出正確的問題，答案永遠都不難找到。

具體範例

我一直都覺得找個實例來看會比較有用，所以，讓我們來看一個情境案例。假設某個人想成為網頁開發人員，利用 Node.js 作為主要開發技術：

目標：成為 Node.js 開發人員

計畫：

學習

- ☐ 學習 JavaScript 的基礎知識。
- ☐ 學習網頁開發技術，例如，HTML 和 CSS。
- ☐ 學習 Node.js 的基礎知識。
- ☐ 具備撰寫簡單 Node.js 網頁應用程式的能力。
- ☐ 學習各種不同的框架和技術，用以開發 Node.js 應用程式。
  - ☐ 根據之前的研究，填入幾個 Node.js 用的框架或技術。
- ☐ 學習和 Node.js 一起搭配使用的資料庫技術。
- ☐ 學習電腦科學的基礎知識：
  - ☐ 演算法
  - ☐ 資料結構
- ☐ 學習以最佳的實務做法寫出好品質的程式碼。
- ☐ 學習如何設計 Node.js 應用程式的架構。

求職前置工作

- ☐ 開始在自己的生活區域裡尋找 Node.js 開發人員的職缺，看看職務說明並且了解雇主需要員工具備哪些技能。
- ☐ 列出我可以從當地哪些公司裡找到這樣的工作。
- ☐ 開始參與這個區域內的同好會。
- ☐ 開始拓展人脈，認識當地其他的 Node.js 開發人員。
- ☐ 聘用履歷寫手，幫我寫一份好履歷。
- ☐ 練習面試當場要寫的筆試問題和程式。
- ☐ 模擬練習面試情境。
- ☐ 建立一份作品集，展示幾個自己開發的應用程式。

開始求職

- ☐ 聯繫人際網路裡的所有聯絡人，讓大家知道我能提供的價值，以及我正在找什麼工作。
- ☐ 開始應徵初階職務或實習計畫的職缺。
- ☐ 規劃自己每天至少應徵兩份職缺。
- ☐ 每次面試結束後，對自己做個簡短的匯報，判斷自己需要掌握哪些技能。

剛開始計畫會比較粗糙，但是隨著你越來越了解自己需要學習哪些內容和做哪些事，你會慢慢填入更多細節。**制定某種計畫並且讓計畫到位非常重要**。你隨時都能改變和調整計畫，但如果一開始就沒有計畫，你只會漫無目的地隨處漂流，或許還會感到挫折，最後很可能就放棄了。

下一章我們會討論為了成為軟體開發人員，你需要哪些技術能力，幫助你進一步完善這個計畫。