關於本書

這並不是一本教科書，這是一本集結了一些能幫助您精鍊思考方式並處理好軟體發展專案之短篇故事的書。

您不會在本書中找到任何整理清楚的制式建議，相對地，您將看到身為務實開發者的我們所面對的許多問題，以及找出問題解決方法的思考過程。

為了鼓勵您以創意發想的方式來閱讀本書，我讓您成為每篇故事的主角。這應該會讓您感到怪怪的，在寫這段介紹時，我自己也覺得怪怪的。

我希望能透過讓您融入到故事情境的方式，讓本書所記錄的不只是一些從編程專家們所住的山峰上，流洩下來的一些嚴肅警語。我要您發問，如“若我遇上這種情況，我真的會用這種方式來處理嗎？如果不會，為什麼？”

閱讀本書時，請您打開內省的聚光燈，深入地質問自己的實務、習慣以及觀點。為了能有最好的效果，閱讀時手邊放本筆記，與朋友及同事分享您記下來的要點。這種想法是要引發進一步的討論，而不是只有單方面的資訊接收。

在每一則故事中，您將扮演不同的角色在想像的世界中遊歷，這些情境是為了傳達有用的經驗而精心建構出來的。其中最重要的部份是要讓您注意到真實的您與我為讀者所建構出來的角色間，所產生的摩擦與具意義的差異。

沒錯，聽來似乎有種雄心壯志的感覺，不過閱讀或編寫一本書的重點不就在於能讓自己有成長，能把事情做得更好嗎？我們現在就站在一起，有您的協助，我想我們會做得不錯。

把安全帶繫上，我的朋友，我們的旅程即將展開。

旅程

本書的軸線涵蓋了軟體開發生涯的各個階段，並將之濃縮成一本小冊子，讓您很快可以讀完，目的在於讓所有想要身體力行的開發者們容易取得其中的要點。

第一章：您是一位有能力的程式員，您想要透過以雛型讓人們瞭解新產品之概念的方式來發揮所長。

第二章：您的工作愈來愈複雜，您需要逐步調整現有系統，也有許多客戶需要支援。一方面要依照您認為正確的方式工作，一方面會面臨到要儘快將新功能上線的壓力。

第三章：您對匆忙決策造成的代價有了深刻的體悟，特別是在程式碼要與外界整合的時間點上。您從過往的錯誤學到了很多，並開始聚焦在業務、客服與技術工作間的複雜關係上。

第四章：您現已是一位經驗豐富的開發者。您有能力協助他人瞭解如何思考編程與問題解決，也開始指導剛踏入這個領域的朋友。

第五章：您已成為一位成功的教師，而且您的開發經驗豐富，即使在即時的演示會中，也能展現個人獨特的思維。您將運用這些技巧在教學上協助學生跨越理論與實務上的斷層。

第六章：您逐漸能掌握全局。您可以指出傳統軟體系統的弱點，並為其設計適當的替代方案，優化業務成果，讓工作流程更人性化。

第七章：您現已熟悉整個軟體產業，能夠在組織內工作，找到各個層面的問題並妥善處理。您的核心競爭力仍在軟體開發上，但您已具備足夠的經驗，能在各個層面上進行有效的溝通。

第八章：您開始懷疑整個資訊產業未來的發展。此時，您可以自在地選擇自己的職涯發展道路。因此，想清楚要往哪裡走以及為何要這樣走，開始成為最重要的問題。

因為軟體開發人員的職涯發展較像繞圈的螺旋線而不像直線，無論您目前的技術水準到達哪個階段，我鼓勵您閱讀本書所有的章節。

我寫的這些故事適用於許多層面，本書也沒有「初級」與「高階」這種分隔線。每一章都是獨立完整的，因此跳著章節讀是可以的…不過為了達到最好的效果，建議您順著章節的安排依序閱讀。

第一章

透過雛型構想專案

設想你正為一個機構工作，協助客戶檢視早期的產品設計與專案規畫。

不管你現正處理的問題為何，第一步都是要找出客戶腦中的想法，然後以最快的速度將想法轉化到現實世界來。對話與框線圖對找起點來說可能有用，但很快地你就需要進行探索式編程（exploratory programming），因為文字與圖表的功用只到此為止。

透過在整個過程中儘早呈現可運行軟體的方式，可讓產品設計變成是一種互動協作（interactive collaboration）。快速回饋迴路（feedback loops）能讓我們儘早發現可能的絆腳石，在它耗掉太多後續開發階段（成本會愈高）的時間與精力之前，將它剷平。

既使在最單純的軟體系統中，也會有許多必要的零件，儘早將它們組裝起來試著跑跑看，可以瞭解它們如何與其它零件進行互動。這是值得做的。就某些方法而言，每一個專案都不一樣，但上述的作法，適用於每一個專案。

這個禮拜，你將與搭檔薩瑪拉（Samara）一起開發一套音樂影片推薦系統的功能雛型（functional prototype）。剛開始的功能集並不需盡善盡美；只要能夠收集到許多對這套產品感興趣之使用者的回饋就可以。

在本章中…

你將學到探索式編程的技巧，它可以讓你在幾個小時內就能建構並提交出有意義的產品概念驗證。

從瞭解專案背後的需求開始

這是套全新的音樂推薦專案，所以你並不瞭解它應該具備什麼樣的功能。你們與客戶羅斯（Ross）坐在一起，討論如何啟始專案的執行：

你：嗨，羅斯！感謝您來跟我們開會。我的搭檔（薩瑪拉）也在這邊。我們已經準備好了，如果可以的話，我們可以開始討論。

羅斯：好，我準備好了。第一步要做什麼？

你：嗯，首先，我們想瞭解您想做音樂推薦系統的原因。瞭解一個想法從何而來的緣由，可以協助我們找出雛型應該要專注在哪些地方。

羅斯：好，當然。幾年來我們一直在部落格（blog）上貼出策劃好的音樂影片名單（curated lists）。我們與一些擅長為不同屬性的音樂建立名單的人士合作，使用者常可透過相關的搜尋，找到我們的貼文。

這些年來，我們在站上分享了超過 4,000 部的影片。這個音樂資料庫還不小，但目前找影片的方式，還是要一個貼文一個貼文地找。

我們開始思考如何讓我們的資料庫能更容易地為使用者來探索。考慮了幾種方法之後，我們認為建構出某種推薦系統，應該是可行的方式。

初版可以比較簡單，不過我們希望能儘快讓幾百位社群中的活躍使用者與部落格參與者，看到一些東西。

你：聽起來，這會是一個很棒的專案！讓我們進一步討論。

對專案背景有基本的瞭解之後，你繼續與羅斯就如何將這些粗略的概念組成一個可行方案談了幾分鐘。通常在專案發展的這個時候，會有一個問題浮現，即眼前的這個專案是要獨立執行還是需要整合到某個現有系統當中。

就這個案子而言，羅斯並不完全確定他要的是什麼。不過，聽到你說這個問題可能稍後再想會比較好，如此才可集中團隊的力量來思考是否能落實這個想法。他贊成。

你提出了幾種可以讓雛型更容易貼近音樂影片部落格閱聽眾的方法，也提出了一個簡單的解決方案：運用部落格本身來搜尋將在雛型中作為範本的影片。如此，新推薦系統中的內容，就羅斯或是其部落格的讀者而言，都是熟悉的內容，而且既使二個系統是在完全分開的碼庫（codebases）上運行，新應用程式與原本的網站間還是會有清楚的連結。

透過框線圖設定預期功能

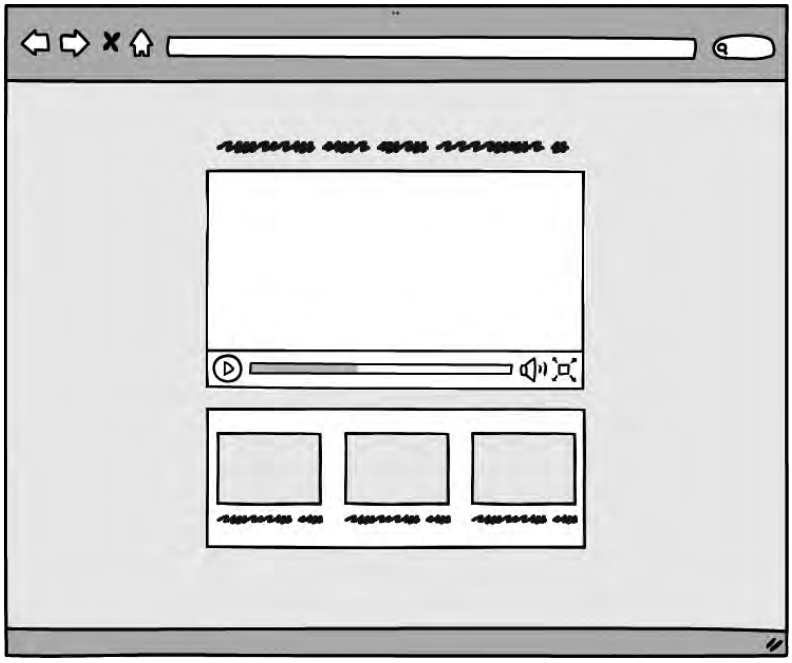
隨著整個大架構的想法愈來愈清晰，你將重點轉到思考如何啟動專案的第一次迭代。在這個階段，框線圖是有用的工具，因為它能讓你呈現出將要建構的基本架構以方便溝通，同時也能讓大家對需要完成的工作產生共識—不會一開始就被實作細節所迷惑。

與其花很長的時間討論推薦系統各種不同的實作方法，你建議由「最單純的可行版本」[1]開始做起。

你：先試著做第一版的基本使用者介面，我們可以從中間有個影片播放器的頁面開始做。可在播放器下方，擺一些推薦影片的縮圖（thumbnail images），這些縮圖是照播放中的影片而挑選出來的。你覺得這樣如何？

羅斯：還不錯，滿合理的。不過我要實際上看到頁面後，才比較能瞭解這樣做好不好。

你：我們在聊的時候，薩瑪拉已經在製作框線圖了，我們可以用它來繼續討論。請等一下，我把圖傳到網路上…



1 這個由 Ward Cunningham（http://pbpbook.com/wardc）所提出的構想，目的在提醒你專注在工作目標上，不要因想像未來的成本或利潤而迷失。

你：你覺得如何？一開始，我們儘量從簡單的做起。

羅斯：很棒啊。很類似我在網路上看到的一些影片播放器，做成這樣的話，我們的使用者也許比較容易理解。

你：太好了！在我們繼續討論下去之前，薩瑪拉跟我會在實際的網頁上做出類似這份草圖的頁面。我們會先用佔位圖（placeholder images）來代表頁面上的元件，所以很快就可以弄好，它有助於測試一些基本的假設，讓我們知道後續的工作應該如何進行。

羅斯：沒問題，如果你認為這樣做有用，那就做吧。

你已經準備好要動工了，但薩瑪拉似乎有些猶豫不前。你問她怎麼了，她說在你請羅斯對介面草稿提出看法時，她剛好想到一個更好的介面。

與其照原本設想好的工作流程，薩瑪拉建議建置一個一次顯示一段影片且帶有「讚（thumbs up）」與「遜（thumbs down）」按鈕的播放器，讓使用者可以表達自己是否喜歡該影片。旁邊可以再安排一個大的「下一段影片」按鈕，按下後可以馬上播放另一段推薦影片。這跟在電視機上轉台類似，不過智慧型系統可以猜你接下來想看什麼。

這個想法很棒，不過實作起來可能沒那麼簡單。經過初步討論並作出取捨（tradeoffs），薩瑪拉接受先從最簡單的開始做起的看法，因為這是讓專案能展示可行性給客戶看的最快途徑。

開始寫碼時即架設即時測試系統

快速雛型的重點是縮短專案中相關人員的距離：不管是在開發人員與客戶之間或者客戶與其顧客之間。

為了滿足這些需求，建造出每一個人可與之互動的系統是非常重要的；它可以讓大家試用而不只是在紙上談兵，也易於讓你分享目前的工作進度。瞭解情況後，你著手進行在互聯網上架構網頁應用程式並讓雛型能夠運行的準備工作。

因為你正使用一套合用的應用程式架設（hosting）平台，通常這代表你正透過常用的網頁製作框架（web framework），設定一個通用的“Hello World”頁面，然後將程式碼上傳到能偵測出所使用工具（toolchain）的 Git 儲存區（repository）上。自此，這個平台會負責安裝所需的相依套件（dependencies）並自動啟動網頁伺服器。

雖然網站的 URL 如 *bady-robot-pants-suit.somehostingprovider.com* 目前看來很怪，但你只用了幾分鐘，應用程式就在互聯網上開始運行了。

到這個階段，你還沒辦法知道是否可以一路使用這個製作環境直到專案完成，但其實你並不擔心這點。你正在寫的是用來取得客戶設定之目標閱聽眾有用回饋的探索性功能（exploratory features），而且在產品完成交付之前，你的程式碼早就功成身退了。

建立基礎架構時，你並不需要完整實作每一項功能—甚至不需要馬上就架設資料庫，因為確切的需求還不明朗。現階段本來就不需要面面俱到，所以你要負責有技巧地將不必要的功能移除。

「我們應該稍微修飾一下外觀嗎？」薩瑪拉問道。

你停頓了一下，想著這個問題，但下一秒你就想到了 YAGNI 法則[2]，這個問題的答案很明確。

「不用。若這個雛型的目的是為了弄出流暢的展示用軟體，供行銷活動上要呈現的螢幕錄影（screencast）之用，我們可能從一開始就要將注意力放在外觀上。不過就這個專案而言，我認為羅斯只是要讓他的幾個朋友看這個雛型，讓他們回饋一些對功能面的看法。最重要的是，因為這是一套簡單的影片播放應用軟體，無論如何，介面應該會相當精簡才對。」

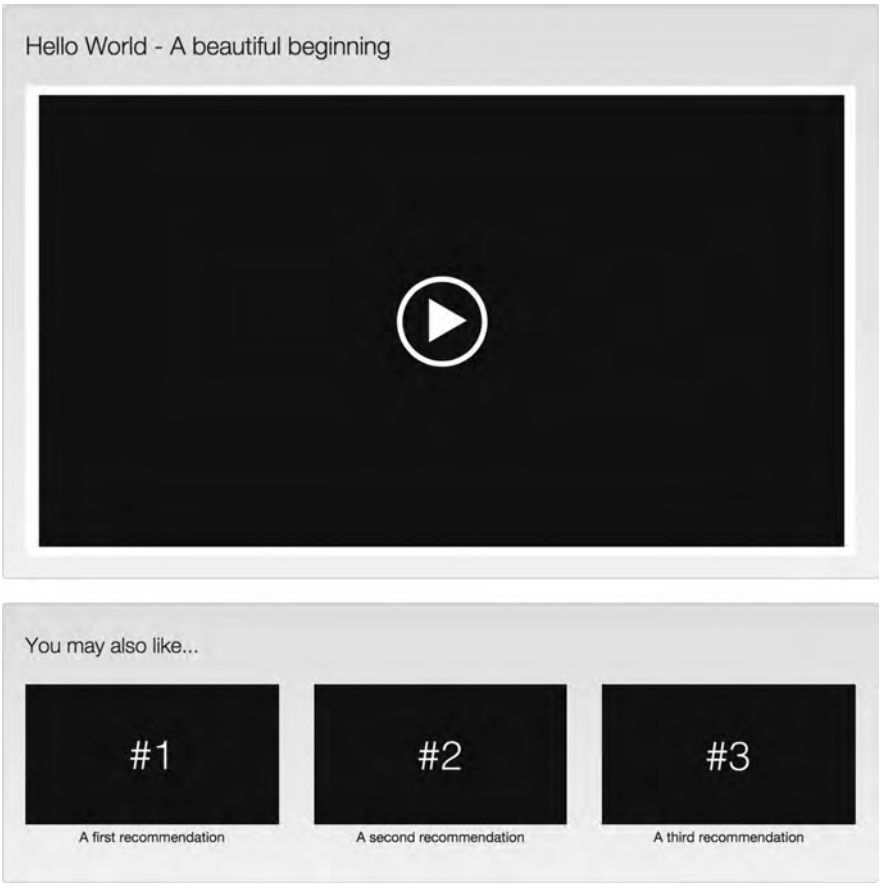
薩瑪拉似乎被你的回答給說服了，既使你只是再一次地選擇權宜的作法而不是看來優雅的方案。不過，你們已經共事夠長的時間了，現在的這種狀況，只是你們習慣的溝通方式而已；薩瑪拉也常常把想太多的你，從許多顧慮中拉回來。

你花了幾分鐘把常用的 CSS 框架接上，薩瑪拉則將一些佔位圖（placeholder）組好。這些弄好了之後，你接著寫一些簡單的 HTML 碼將這些影像排到分格版面（grid layout）上，影像帶有寫好（hardcoded）的標題。

你為了將歌曲標題排好並定好其大小，用了稍長的時間。但你很快地意識到二個重點：目前這些細節根本不重要，而且應該要讓某些東西先跑起來再說！

你將目前寫好的碼不加修飾地佈署出去，一下子網頁就活靈活現地出現在網路上了：

2 你並不需要它（You aren't gonna need it，YAGNI）（*http://pbpbook.com/yagni*）—聲稱功能在真正需要之前才應該被加進來的設計法則。



網頁目前看來並沒有什麼特別之處，你開始擔心客戶能不能瞭解為何你要給他看這麼陽春的東西。

為了檢查你的假設，你問薩瑪拉對目前網頁的看法。她認為既便是最簡單的事也應該在第一時間與客戶溝通，提早詢問客戶的意見總比讓客戶等在那邊要好。

比較確定後，你提醒自己第一次迭代的目標是架設好一個可運行的系統，能夠很快地在其上佈署新的版本，並且啟動整個探索專案的過程。自此，客戶可以直接與軟體互動，這也能加快專案的進行。

你傳給羅斯一個短訊，讓他知道你有進度要讓他檢查，在收到他的回應前，你可以利用這個空檔，稍事休息。

討論所有缺陷務實面對所需的調整

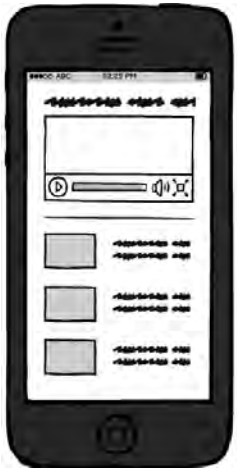
回到辦公桌前，羅斯已經回訊給你了：

> 開發團隊你們好！
>
> 我剛試了一下，網頁在我的筆電上看來，與薩瑪拉畫的草圖差不多，這樣應該可以。
>
> 我也用手機試了一下，但網頁看來就有點奇怪。這些影片是滿版的，推薦影片被顯示成一長串，而不是並排在一列上。
>
> 我們不需要很快地就要讓網頁看來很美觀，不過，我們希望要讓所有推薦影片可以列在同一個畫面上，而不是要滑過一些滿版影片才能找到。
>
> 你有什麼看法嗎？
>
> —羅斯

如薩瑪拉之前所預料的，既使像是這麼簡單的首版，也會有一些問題。你不可能一次就做好，但重要的是你如何處理這些問題。

你知道行動裝置 UI 的問題是迴避不了的，但你也不想就卡在這個地方。比較恰當的回應也許是再重新畫一張網頁的草圖給客戶看。

薩瑪拉開始用手機連到幾家比較受歡迎的影片網站上，觀摩幾種常見的版面設計。然後她畫了一些類似的頁面，去除較不重要的外觀裝飾，保留基本的頁面版型。



你將這張線框圖寄給羅斯，並花幾分鐘跟他討論下一步要做什麼：

羅斯：謝謝你們畫的草圖！看起來它會是一個很好的起點。

你：很高興你認可它。現在我們需要決定：要馬上修正手機版的畫面還是先將它擱著之後再做？

薩瑪拉跟我都覺得應該要先做一些有用的推薦功能出來，然後再一路修改 UI 下去。

也就是說，若你覺得手機版的設計在早期收集回饋階段就很重要，我們可以先花一點時間把這部份處理好後再繼續。

羅斯：相較於現在就處理，之後再處理會不會多衍生出一些額外的問題？

你：我覺得應該不會。這套推薦系統大部份的工作都會先串好，因此電腦版的 UI 應該不需要有太大變動。若因為某些原因需要調整主要的 UI，我們還是需要先畫出手機版的草圖。

羅斯：好，那我們就緩一緩。不過，若有先期試用者回應沒辦法方便地透過手機來操作系統，也許我會改變主意，目前我們可以先等到開始收集回饋後，再來操心這個問題。

找到軟體中的問題時，你可能馬上就放下手邊的工作去處理這個問題。不過，在專案的探索階段，重要的是如何在每一個問題成本與該問題處理成本間，取得平衡。

在這個範例中，不用在處理行動版樣式小部份調整的時間，就可以用來探究音樂影片的資料集並瞭解推薦的規則。不過，讓客戶瞭解你將如何處理未來可能遇到的問題，也有助於事先避免在小事上投注過多精力的風險。

儘早且經常檢測假設

與羅斯進行這些先期溝通後，你大概知道他只是想為他的音樂社群創造出一些樂趣，比起「打造世界上最複雜的音樂播放服務！」或類似的目標，要單純很多。

不過，我們還是需要儘早檢驗對一些事情的假設。初版草圖著重在應用軟體的 UI，現在應該要開始討論它的運作方式了：

你：接下來我要請教比較技術性的問題…

我們要實作的推薦方法有什麼規則？

羅斯：噢…嗯…我原本希望能瞭解你們對這方面的看法。幾週前，我們都還不確定這個專案是不是有執行的價值，目前我們還沒有對這方面進行深入的研究。

你：其實這個部份有許多選項可採行，從最簡單的比對到運用機器學習的複雜方法都有。要採行何種方法視現實條件而定，雖然不管用哪一種方法我們都能幫你開始進行開發，但這個領域並不是我們所專精的。

羅斯：我不確定這能不能讓你們比較容易瞭解我們的需求，不過，我們目前的部落格都是以列表方式來呈現的（比方說，「十首您可能從未聽過的麥爾戴維斯（Miles Davis）作品」、「1980 年代的紐約現場嘻哈表演選集」或「適合家庭聚會時播放的聖誕歌曲」）。

我們希望這個推薦工具能協助閱聽眾不受到這些列表的限制，能在不同列表中找到他們可能會喜歡聽的作品。因此，比方說，他們可能正聽著一首 1980 年代在紐約現場演唱的嘻哈舞曲，我們就可以幫他們找到相同樂團或歌手的作品，或是其他 1980 年代的嘻哈舞曲等。

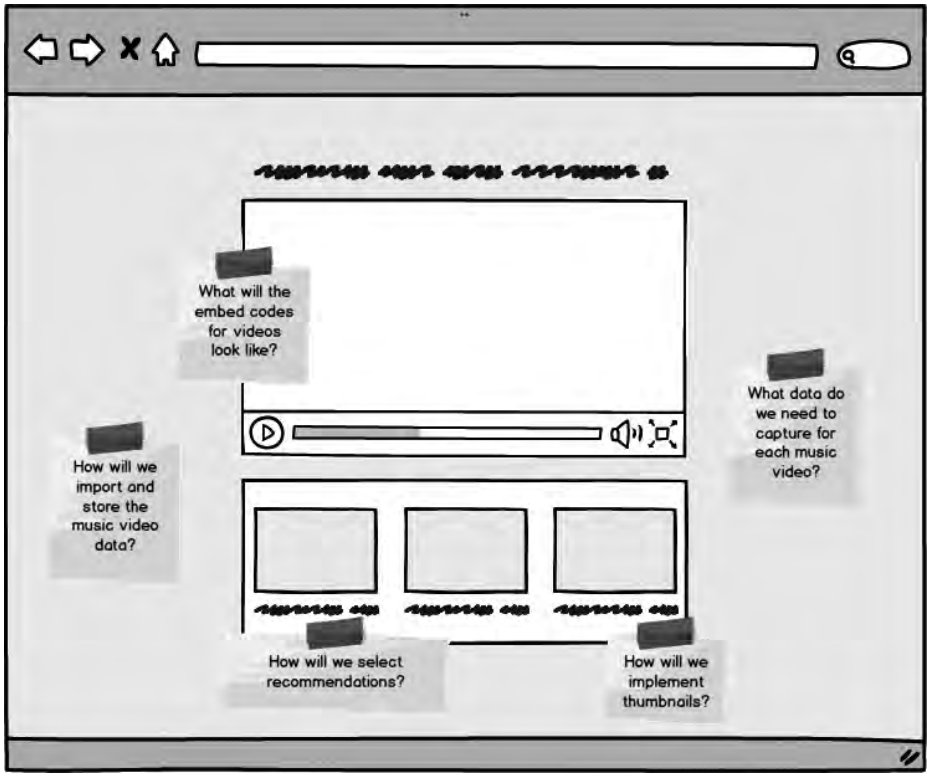
你：OK，我們可以從這方面來思考，謝謝。我想薩瑪拉跟我可能要用一點時間來消化這些東西，明天我們再讓你看一些想法，這樣好嗎？

羅斯：太好了！謝謝你們的幫忙；跟你們一起討論滿有趣的。

上述的討論確認了這個案子應該只需要一個簡單的推薦系統，而且羅斯對實作的細節似乎沒有太多的堅持。這個專案做起來應該會很順利，不過如果他對專案還存有更複雜的想法，早一點把這些想法挖掘出來會比較好。多提問無傷大雅！

儘可能為工作設立範圍

至此所做的所有事都只是為了要找尋專案的切入點，現在你可以捲起袖子，做一些正事了。目前還有一些未知數要探究清楚，光研究薩瑪拉所畫的原始草圖幾分鐘，就可以找出一些關於實作細節的問題：



這些問題並非依序出現，但在你開始動手之前，你需要為它們安排優先順序。

就你與薩瑪拉所找到的 5 個重要問題而言，有 2 個看來比較簡單：如何為這些影片產生內嵌碼（embed codes），及如何產生縮圖（thumbnail images）的 URL。

你連上羅斯的部落格播放其中的音樂，想要找出提供這些影片的所使用的服務。你也點進了幾條貼文，檢視其原始碼以瞭解其中的結構。

「大部份貼文所內嵌的影片是使用 FancyVideoService 的服務。內嵌碼遵循標準的格式；每段影片間用其專屬的識別碼（identifier）來區別。

「縮圖呢？這些縮圖有沒有什麼玄機？」

你遲疑了一下，然後更密集地點部落格貼文中的連結，一直到你沒有再找到沒看過的結構。

「看起來他們的網站實際上並不是使用縮圖，目前我看到的都是嵌入影片，我想我們要把這些縮圖找出來才行。」

你花了幾分鐘在互聯網上搜尋，不過並沒有找到任何關於如何擷取 FancyVideoService 影片縮圖的官方說明。不過你還是找到了一則部落格貼文，說明他們內部使用的 URL 格式，很容易透過與影片嵌入碼所使用之相同的唯一識別碼，來產生縮圖的連結。

根據羅斯部落格上的影片，你手動製作了幾個縮圖的 URL，雖然還不確認這種方式是不是原廠所支援的使用案例（use case），但縮圖是可正常顯示的。目前看來雖沒有問題，但你還是必須要在專案完成前與 FancyVideoService 聯絡，以確認這種方法是官方所允許的存取方法。

排除了這些瑣事，你可以回到檢視模版（mockup）時關注的重要問題上：要收集這些音樂影片的什麼資料，如何儲存這些資料，以及如何使用這些資料來產生有用的推薦。

你與薩瑪拉開始討論一些方法，不過，很快地就意識到討論的焦點愈來愈偏離重點。於是你回到那個最經典的問題上：「最小可行版本是什麼？」

沈思了一會兒，薩瑪拉的靈感冒出來了。

「我們要不要從比對演出者來著手？根據正在播放的影片，隨機選幾首由相同演出者所表演的歌曲來呈現。」

「好主意。我想在羅斯把雛型送出去收集回饋前，我們還需要想出一些比較複雜的方法，不過目前我真正需要的是能呈現在螢幕上，供我們互動討論的東西。」

表演者比對是簡單的起點，因為這個功能所需的材料不外乎是 FancyVideoService 的影片識別碼、歌曲名稱以及表演者的名稱而已。若你從羅斯的部落格上抓出一些歌曲，就可以形成足敷運用的資料集。

「我們應該怎麼處理資料的儲存？我應該提前預防一些…」

薩瑪拉為了讓事情保持單純，突然打斷你的話。

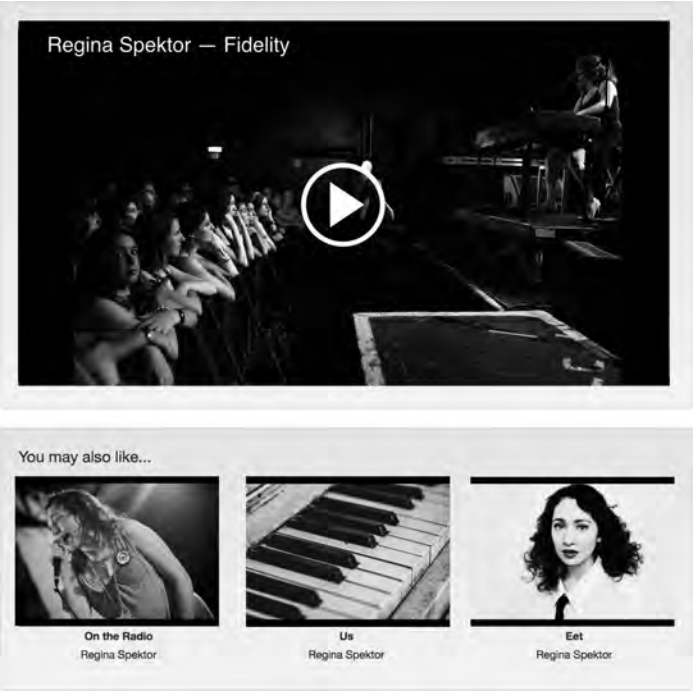
「目前還不需要想那些。我們可以先用一些陣列，直接寫出一個簡單的資料集，然後透過這些資料來產生推薦。」

「你知道那樣子做撐不了多久，對吧？」

「沒關係，它也不需要撐很久。」

薩瑪拉已經準備好了，所以你要她在你整理歌曲名稱、演出者與影片識別碼時，寫一些程式。

15 分鐘之後，你們二個就在線上弄出看起來有模有樣的半成品[3]了。



為了呈現目前的進度與未來的走向，在結束今天的工作前，你傳了今日最後的紀錄給羅斯：

> **你：**嘿，羅斯，你現在連上網站就可以看到我們做了一些看來像音樂推薦系統的東西。雖然目前功能有限（只能使用表演者比對功能），但我們覺得你應該會對我們目前將一些元素組合在一起的結果，很感興趣才對。
>
> **羅斯：**哇噢！做得很棒。比起只能與臨時代用影像互動來，這感覺好多了，它看來似乎已把我心裡想要做的大部份東西都實現了。

3 影像來源：Piano（*http://pbpbook.com/piano*）；Regina #1（*http://pbpbook.com/reg1*）；Regina #2（*http://pbpbook.com/reg2*）；Regina #3（*http://pbpbook.com/reg3*）

我覺得你們會在明天把更多有趣的推薦方式加進來對不對？這個系統並不需要太潮（fancy），不過，多一些推薦方式的選擇，而不是只有表演者比對的推薦方式，當然更好。

你：沒錯。我們仍持續在思考其他的方式，不過要明天才能讓你看進度。

羅斯：太好了。再次謝謝二位。能在一個工作天內就看到想法被實作成初版的雛型，我實在太高興了。

你與薩瑪拉架構出來的基本**運行骨架**（*walking skeleton*）[4]，在未來的迭代中，將愈來愈豐富有趣。

當工作第一天的大部份時間都用來將零件組裝好就定位後，你已取得開始探究真正問題的先機。如果你直接就從思考如何解決整個問題著手，可能比較難以找到著力點（starting point），整個過程中也會遭遇到許多障礙。

今天的時間所剩不多，你決定用下午的時間來處理一些瑣事、瀏覽部落格貼文，也透過手機上網抓寶，以換取一些互聯網上的點數。

切記雛型並非產品系統

就這樣安靜地過了半個小時，薩瑪拉帶來了令人振奮的消息，打破了沈寂：

「嘿，FancyVideoService 的客戶部門傳回應給我們了。」

「真的嗎？我不知道你有寄電郵給他們，你哪時候寄的？」

「你跟羅斯在討論的時候寄的。我想早一點把這點弄清楚比較好，不過，這麼快就收到回覆，我也很訝異。」

你站她的座位後，從她肩膀上的空隙往電腦螢幕上看：

嗨，薩瑪拉：

因為我們要分享影片，要能支援較多的使用案例，就技術上而言，直接連結到我們站上影片的縮圖，並沒有違反我們的規定。

4 運行骨架（*http://pbpbook.com/skel*）是對一項功能的一組小型流程（end-to-end）實作，它是一個起點，讓你思考涵蓋它的系統架構。

不過，我們沒有為這種存取方式提供支援，也不保證 URL 的結構不會變動。我們亦保留拒絕不當使用這項服務之存取操作的權力，我們會自行判斷並為適當之處置。

比較好的作法是在我們的開發者社群下註冊，然後使用我們提供的資料存取 API。透過這種方式來查找縮圖的 URL，既使在未來我們的 URL 結構有更新，你的程式碼仍可正常運作。

註冊開發者社群的另一項好處是，若你的程式碼不小心違反了我們的服務條款，我們會通知你並說明如何處理這類問題。

希望上述說明有助於解決你的問題，祝你有個「美好」的一天！

—莎拉

知道官方允許開發者使用這個功能後，你鬆了一口氣，既使現在的使用方式並不是 FancyVideoService 建議的。

為了節省時間，你決定先以這種非官方公開支援的方式來產生縮圖的連結，不過你也將這個問題記錄下來，讓負責產品正式版的開發人員知道有這件事。

你很滿意今天的進度，圓滿的一天。

設計容易收集回饋的功能

隔天早上，你一進辦公室就看到白板上寫滿了昨晚你離開時還沒有的筆記。你很好奇薩瑪拉做了些什麼，你開始看白板上的筆記。

「喔，哇噢！羅斯會愛死這個。早上妳幾點進辦公室的？」

「約一個小時前。吃早餐時想到這個點子，我就趕快進辦公室先來玩看看。」

薩瑪拉看來像是睡眠不足，但因為你很喜歡這個點子，你也就沒有問她昨晚幾點睡覺。

「嗯，我們要開始做這個嗎？初步看來很不錯，你寫的那些非常非常棒。」

「已經做好了，請看網站。」

你坐回椅子上，花一些時間玩玩剛寫好的新功能。這些功能都運作得很好，就第一次要交付的雛型而言，已經很好了。

「妳怎麼這麼快就把它做好了？我原本就以為妳會像我們之前所做的那樣，把功能修整一下，沒想到妳在一個小時內就做了這麼多事。」

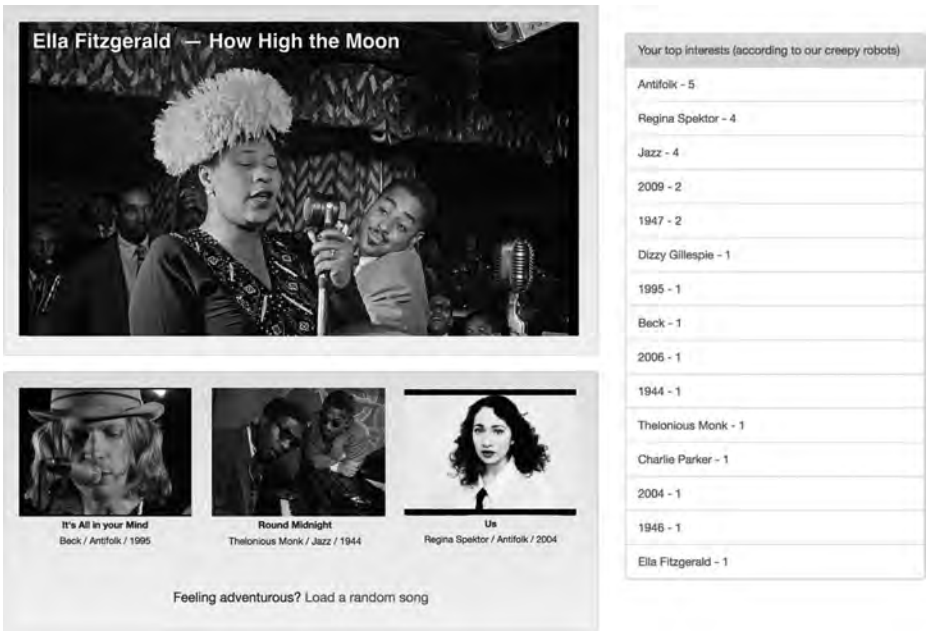
「喔！你應該不會想要看這些程式碼。看到那些推薦分數了嗎？它們全部存放在一個瀏覽器的伺服器紀錄（cookie）裡頭。」

要能拿捏出專案何時該快該鬆，何時該紮實地進行需要練習，目前你信任薩瑪拉的判斷。你透過網路戳了羅斯一下，想知道他對新功能的看法：

你：嗨！羅斯。我們這邊上了新的版本，麻煩你有時間檢查看看。只要連到網站上就可以看得到，我可以逐項說明這些功能。

羅斯：我會儘快利用時間看。謝謝。本來我想說你會在中午左右才會有消息，這真是一個令人愉快的驚喜。

你：這些是螢幕截圖[5]，呈現出瀏覽一些影片之後的頁面，你一定要親自操作看看，才能完整體驗其效果。:-)



5 影像來源：Ella Fitzgerald（*http://pbpbook.com/ella*）；Beck（*http://pbpbook.com/beck*）；Thelonious Monk（*http://pbpbook.com/monk*）；Regina Spektor（*http://pbpbook.com/reg2*）

羅斯：我剛用幾分鐘試了新功能。太棒了！

我看到頁面上多了「感興趣」側欄，我們昨天並沒有討論到這個。你可以說明一下這個功能的用途嗎？

你：沒問題。在說明其用途前，我先要讓你知道，這個側欄並不會一直出現在應用程式的操作介面上。

因為要解釋推薦行為比直接試用這個功能要稍微難解釋一點，所以我們在頁面上就做了這個側欄，讓你可以看到標籤分數（tag scores）如何加到所選的影片上。每一個標籤都可點按，當你點按了一個標籤後，系統就會將所選類別中隨機挑出的影片帶出來給你。你可以透過這種方式來影響分數，以及調整推薦行為。

羅斯：你可以給我一個操作過程的範例嗎？

你：沒問題。請點按幾次“Thelonious Monk”，然後看看會有什麼變化。

羅斯：啊！我點了之後，民謠龐克（antifolk）風的音樂變得愈來愈不受推薦，而爵士（jazz）風音樂被推薦的頻率變高了。我想一直點到最後，系統應該除了 Monk 的影片之外，就不再推薦其他風格的音樂給我了。

你：沒錯，現在你應該比較瞭解推薦系統雛型的運行方式了吧？

羅斯：我想這樣應該能讓我玩一陣子了，而且今天也應該能把雛型傳給其他人玩玩看，然後收集他們對雛型的意見。

之前因為推薦系統的運作方式，光只聽你用講的我不太能瞭解，不過，我**真的**很高興你們做出這個側欄，讓我能更快地瞭解它。很謝謝你們。

你：這是薩瑪拉的主意，我們會把它弄得更好。它可以讓你先瞭解一些系統運作的機制，看看我們實作出的一些規則。

***

羅斯：在將它送出去給人試用之前，我還有一個問題要問你：這些樣本的資料是從何處取得？

你：目前這些樣本資料是從你的部落格上，隨意挑選出來用的，之後我們會做個能讓你自行設定的機制。

系統目前是從一個 CSV 檔中，讀資料出來用的，你可以用試算表軟體來編輯這個檔。底下所列的是目前系統所使用的部份資料。

q97xzziKqOI	Charlie Parker	Chasin' the bird	Jazz	1947
zre0u5XyNfY	Thelonious Monk	Round Midnight	Jazz	1944
oslMFOeFoLI	Dizzy Gillespie	Groovin' High	Jazz	1946
9KwLWpU0_K0	Ella Fitzgerald	How High the Moon	Jazz	1947
tHAhnJbGy9M	Regina Spektor	On the Radio	Antifolk	2006
fczPlmz-Vug	Regina Spektor	Us	Antifolk	2004
MMEpaVL_WsU	Regina Spektor	Eet	Antifolk	2009
4RJob0jSCX4	Regina Spektor	Dance Anthem of the 80s	Antifolk	2009
Z6XiO0o2R7M	Beck	It's All in your Mind	Antifolk	1995

你：第一欄（column）是影片的唯一識別碼，它會出現在每一項 FancyVideoService URL 的最後面。第二欄是演出者的姓名，第三欄則是曲名。之後的每一欄可視為自行定義的標籤（tag）。目前我們只有 2 個標籤（分類與發行年），不過，你可以再加進需要的標籤，數量不受限制。

羅斯：等等…不知道我想的對不對？如果你將這份試算表傳給我，然後我在裡頭加進我要的影片資料，你就可以將它直接匯入（import），系統裡頭就會有這段影片，而且還帶有我設定的標籤？

你：沒錯。初步的想法是如此。剛開始處理時，我們可能要小心一點，因為這些資料的格式要正確設定好，運作起來才不會出問題。不過這方面若有需要的話，我們可以協助你正確地設定好資料。

目前我們需要請你從部落格的現有資料中，擷取出幾百首歌曲的資料，並產生列表，如此，我們就可以透過現行的資料來測試推薦系統的運作。

你提供上述資料後，我們再來研究看如何自動地將部落格上約 4,000 首歌的資料，轉到系統裡頭。這部份的工作，可以先緩一緩，稍後再來做。

羅斯：太棒了。我會儘快照你建議的來做，從部落格上整理一份列表來。之後，我會找一些熱心的網友，請他們今天就連上去試用雛型，今天晚上前，應該就可以傳一些回饋給你。我們就可以依據他們的回饋來調整。

很感謝你把這些都整合好了，做得很棒。

你：這個案子很有趣，你也幫我們很多忙，我們也感謝你。

儘管目前雙方都覺得滿意，但並不代表專案之後的發展就會一路順遂。如古諺所言，「魔鬼藏在細節裡」，在接下來的幾次迭代中，需要處理的細節會愈來愈多。在完成雛型階段前，可能會遭遇到預期外的重大問題。

不過這並不是說目前流程有重大的缺陷；你應該要將預期外的問題視為快速回饋迴路的副作用。雛型能協助你很快地建造出有用的東西，但它也會讓你很快地失敗。如果在花時間往死胡同鑽之前，你就能看到這條路行不通，表示你可以將精神專注在找正確的路上。

不論如何，你與薩瑪拉現在應該可為專案初期的順利進展而高興。專案發展初期所建立的善意與信任，有助於產生動能，讓你在進行創意工作中遭遇困難時，能順利地將之排除。

建議與提醒

- 提出可以揭示專案參與人員目標的問題。如此，不但可以驗證假設，也可以更瞭解相關人員對問題的看法。
- 透過線框圖（草圖）清楚地用應用程式的基本結構進行溝通，如此就不會陷入太多風格設計方面的細節。
- 開始寫碼時，就建立所有人都可與之互動的即時測試系統（live test system），初期的系統並不需要馬上就能上線運作；只要能收集到有用的回饋即可。
- 在專案的初期，要專注在工作中風險高或未知的部份。雛型可用來找出可能的問題，但它並不是最終的系統。

問題與練習

Q1：本章的音樂推薦系統開發工作，目前進行得相當順利。有沒有什麼可能做錯的事情（實際上沒有），會讓開發者不好處理的？

Q2：選出二項開發者為了貪圖一時方便而閃躲的工作。思考作這種決定時會需要有何種取捨？換言之，開發者犧牲了什麼來加快一點點進度？

Q3：設想看看，若客戶想要的是運用機器學習技術實作之較複雜的推薦系統，就專案雛型階段的發展方向而言，會有什麼影響？

E1：畫一些線框圖說明一個維基（wiki）網站的基本功能。接著重複整個流程，但畫出不一樣的介面。想想這二種實作細節間有什麼差異。

E2：拿你目前所使用的任何軟體工具或網站為例，假設你要從無到有將它實作出來。用個把小時的時間，想想第一步要怎麼開始。

嘿，你已完成第一章！做得很好。

為了感謝你所投入的努力，請享用這個與本章主題不相關的解迷遊戲[6]。

>	22	#	6F	!	>	AF	#	CA	#	34	>	A9	>	A2	00
00	00	>	A1	00	00	#	34	00	42	42	00	42	FA	F2	FE
?	21	#	68	>	57	42	3D	FA	F2	FE	42	3D	87	00	87
00	00	>	A1	00	00	#	34	00	00	3D	FA	F2	FE	00	87
42	3D	FA	F2	FE	?	5A	00	87	?	17	00	87	FA	F2	FE
00	00	3D	3D	3D	00	42	00	3D	FA	F2	FE	87	#	00	?
42	3D	FA	F2	FE	42	3D	00	00	>	A1	00	00	#	34	00
31	21	00	21	21	#	65	#	6C	>	CC	21	FA	F2	FE	45
?	FA	F2	FE	?	02	00	00	>	A1	00	00	#	34	00	45
31	31	00	FA	F2	FE	?	17	FA	F2	FE	21	21	00	00	45
31	FA	F2	FE	00	00	00	00	>	A1	00	00	#	34	00	87
?	31	00	?	02	00	00	00	>	A1	00	00	#	34	00	FA
31	?	31	FA	00	FE	00	00	?	17	00	FA	#	6C	>	20
31	#	00	00	>	A1	00	00	#	34	00	#	FA	FA	F2	FE
FA	F2	FE	00	FA	00	00	?	17	00	FA	F2	FE	CF	?	FA
FA	F2	FE	00	00	>	A1	00	00	#	34	00	FA	00	?	00

從頭開始，碰一聲就結束！必要時要跳來跳去，但別被這些干擾搞亂了。仔細觀察，你一定可以拼湊出隱藏在背後的訊息。

6 你不需要編寫程式來解這個問題，不過找張 ASCII 表來會比較方便。瞭解這個小圖靈泥沼（little Turing tarpit）背後的運行規則後，用紙筆就可在幾秒鐘內解開這個迷圍。