
作者序

2005 年三月，我又驚又喜地收到通知，我寫的一本書「Better, Faster, Lighter Java」（中文版《輕快的好 Java》，歐萊禮出版）贏得 Jolt Award 技術卓越獎。我在那本書中告訴 Java 開發者如何建立更好的程式，比以往更快的程式。那本書在我心中的特殊地位將永遠不會被取代，但是，這一路下來，有些東西出現了，我必須動手去接觸。

在這段期間，我的一個顧客正在建立一個應用，包含複雜資料庫綱要（schema），以及 web 使用介面，我們用了 Spring 和 Hibernate 搭配 Web Work。對於輕量級的 Java 開發來說，這是很常見的組合，我們一般也算滿意。但是有些事情在困擾著我們：重複的事情太多、XML 組態（configuration）激增、改變的步調太快。一時興起，我們試了 Ruby on Rails，這是一個高生產力、創新的框架，正在席捲非 Java 的社群，而且也為 Java 的架構帶來一些爭論。使用 Rails 所獲得的高生產力，讓我們感到震驚，所以後來我們把整個計畫都搬過去了。

有些東西和我一拍即合。對於這種應用來說，Java 是個礙手礙腳的東西。將 Java 排除不用，我可以把程式碼減少到只剩四分之一，把 XML 減少到只剩十分之一，達到令人驚訝的生產力，以及不錯的效能。更棒的是，「Better, Faster, Lighter Java」（簡稱 BFLJ）書中所提到的概念，依然可以在此採用。對於其他的計畫，如果我需要 Java 所提供的社群和工具，我用它來取代；如果我不需要 Java，我可以把 BFLJ 的原則發揮到極致。我內心的水壩已經潰堤，藉由這本書傾洩而出，我要告訴大家我的想法。

幾個月後，我找到聽眾了，在數千里外，離我家有 19 個小時的路程。在 Java 使用者社群的前面，我感到侷促不安，我在更大的場面中演講過，但是這趟旅程卻很不一樣。這次挪威 Java 使用者社群幫我支出到奧斯陸（挪

威首都) 之行的花費，他們卻無法在會場中聽到我歌頌 Spring、讚揚 Hibernate、褒獎敏捷開發 (Agile Development)，而是會聽到我說出「他們摯愛的寶貝很醜」，我是否要實話實說，我陷入天人交戰。Java 為我帶來許多好處：寫了暢銷書，得到 Jolt 獎，在不佳經濟局勢中擁有相當不錯的顧問業務，我當然希望 Java 列車繼續開下去，不會停歇，永遠根基穩固，我希望 Java 讓我的生產力直上雲霄，無盡的群眾和腦力會解決 Java 今日所面對所有棘手的問題…但是世間沒有任何東西可以維持永恆。

在我的演說中，我的確挑了一個最終的贏家。我先陳述 Java 成功的理由，然後論及它嚴重的問題，我展示一些另類的語言和框架。在整個演說中，我指出時機已經成熟，該是另一個技術出現的時候了。當我對這群好客的人演說時，我回答問題，也觀察人的表情，一些人看起來具有敵意，有的人則是很受傷，但是大部分的人則是很能理解，或者有一些恐懼。他們瞭解我的中心想法。對於我們慣用 Java 來做的事，有些其他語言的框架做得更好。在某些狀況下，生產力的提升實在太大，值得我們好好地研究新的技術。

這場演說以及問答超出時間許多，但是沒有人離席，令我驚訝地，他們都能接受。在演說完畢後，我們出去看看奧斯陸，一個具有敵意的聽眾幾乎整晚都圍繞著我，拋出一個又一個困難的問題：

- 為何我們不能改變 Java，以改正這些缺失？
- 你所展示的其他語言和框架，有沒有足夠的商業支援？
- 它們有支援分散式交易、Web Service、XML 嗎？
- 怎麼找到會這種技術的編程員 (programmer)？又該如何訓練自己的編程員？

這些問題都是真正需要注意的問題，這些問題也顯示出進入新語言時必須跨越重重藩籬。問問題的人是個紳士，但是他無法完全隱藏自己煩躁不安的情緒，以及根深蒂固的思想，他認為進入下一個成功的語言需要付出相當高的代價，他也認為在可見的未來我們還是會繼續在 Java 上面寫程式。他可能沒錯，但是我已經開始發現 Java 語言的限制和許多框架的缺失。在某些問題的解決上，Java 已經不再具有高生產力了，我改用一些其他的技術，且體驗到成功。雖然語言可以存活大半世紀，以支援舊系統，但是我知道沒有語言可以永遠領先不墜，Java 統治的時代將會結束，這不是「會不會」的問題，而是「什麼時候」的問題。

誰需要讀這本書？

當 C++ 褪色不再讓人注意，我的許多好友都傷得很重。他們沒注意到，空氣中已經有了不同的氣息，巨大的改變瞬間來到。雖然我寫這本書會失去許多，但是我寫這本書的目的，是希望這樣的事情不要再發生。如果你不希望在一覺醒來之後忽然發現自己已經被淘汰了，那麼你需要讀這本書。

如果你認同我的說法，你可以開始試著建立你自己的技巧。你可以下載一些我討論到的框架，學習一些語言，這本書會告訴你，成功的語言所具備的要點。如果我很幸運，挑出一些未來的贏家，那麼以後改朝換代時，你就不會措手不及了。

如果你不認同我的說法，你可以用最好的語言，最好的框架，最好的技術，來改進你今日用 Java 在做的事。新的框架像是 PHP、C Omega for .NET、Ruby on Rails 將會偶而出現在你的周遭，你需要知道它們，瞭解如何評估它們。

所以，不管你認同我的說法與否，你都是贏家。該是注意的時候了，該是看著水平線的時候了，看得比 Java 更遠，超越 Java。

字型慣例

由於本書保留了部份術語的原貌，為了避免讀者的混淆，我們運用了一些字型變化，讓讀者能輕易辨別詞彙本身的含意。

定寬 (**constant width**) 字型：

- 任何在程式碼中可能出現的元件，包括關鍵字、運算子、資料型態、函式名稱、變數名稱、類別名稱和界面名稱。
- 在 HTML 文件中出現的標籤 (tag)。

斜體字型 (*Italic*)：

- 用於可被替代的部份。例如：

protocol://www.host.com:port/file#ref

代表其中的 protocol、host、port、file、ref 必須代換成實際有意義的值。

粗體定寬字 (**constant bold**)

- 出現在螢幕上的字樣。為了區分手動鍵入的文字，以及程式自行產生的輸出訊息，前者會以粗體定寬字 (constant bold) 的樣式來表示，後者則是一般的定寬字型。

- 專有名詞。某些重要的術語在第一次出現時，會將其中文譯詞、英文全名、縮寫（不一定三者全部都有）用下列格式表示：

檔案傳輸協定（File Transfer Protocol，FTP）

某些有縮寫的術語，在後文再次出現時，會視情況以縮寫出現，不一定會使用較長的中文譯詞。此外，若是在內文中列舉多個項目時，為了顯目起見，會以粗體字來表示。

使用程式碼範例

幫助你完成工作是本書的目。一般來說，你可以在你自己的程式和文件中使用本書的程式碼。你不需要先聯繫歐萊禮公司，除非你明顯地進行重製的行為。比方說，寫程式的時候使用到本書的片段並不需要我們的許可，銷售本書的範例光碟就需要我們的許可才行。回答問題時引用書中的範例，也不需要我們的許可，但是在你的文件中大量使用本書的程式碼就需要我們的許可。

線上資源

與本書相關的資訊，包括範例程式碼、相關資源連結、勘誤表，皆可在本書的中文網頁取得：

http://www.oreilly.com.tw/product_java.php?id=a187

批評和建議

我們永遠樂意聽到讀者對出版品的意見，包括如何讓本書可以更好的建議、指正本書的錯誤、或是讀者建議本書往後改版時，應該再加進來的其它主題。以下是本公司的聯絡資料：

美商歐萊禮股份有限公司台灣分公司

電話：(02) 2709-9669

傳真：(02) 2703-8802

Web：<http://www.oreilly.com.tw>

電子郵件：

sales@oreilly.com.tw

（業務部）

edit@oreilly.com.tw

（編輯部）

bookquestion@oreilly.com.tw

（書籍內容的問題）

請以電子郵件的方式與我們聯絡，這會比電話和傳統郵件方便。有興趣為本公司翻譯書籍的人士，可與編輯部聯絡；如果您買到的書有印刷品質上的問題，可以寫信到業務部；若您對書籍內容有疑義，或是發現錯字，請寫信到 bookquestion@oreilly.com.tw 與我們聯絡，謝謝您！