序

Java 近 10 年來的發展令人驚嘆。儘管 Java 的聲勢快速提升，不過 Internet 的成長甚至更為突出。讓人有點意外的是，仍有許多人認為「用 Java 進行網路程式設計」是件不可思議的事。事實並非如此，因為用 Java 寫出網路程式非常容易，而本書將證明給讀者看。曾經在 Unix、Windows 或 Macintosh 環境中寫過網路程式的讀者，應該會驚喜地發現，用 Java 寫出等效的網路程式更為簡單。因為 Java 的 core API 為大部分的網路功能提供了完善的介面。事實上，凡是能用 C/C++ 寫出來的應用層網路軟體，多半都能用 Java 輕鬆寫出來。《Java 網路程式設計・第 3 版》將告訴你如何發揮 Java 網路類別庫的優點，讓你輕鬆快速地完成許多常見的網路功能。這些功能包括：

- 透過 HTTP 瀏覽 Web
- HTML 的剖析與呈現
- 透過 SMTP 發送電子郵件
- 透過 POP 和 IMAP 接收電子郵件
- 設計多緒伺服器
- 為瀏覽器安裝新的協定和內容處理器
- 對秘密通訊進行加密、身分確認以及保證訊息的完整性
- 為網路服務設計 GUI 用戶端
- 上傳資料給伺服端程式
- 使用 DNS 查找主機
- 透過匿名 FTP 下載檔案

- 為低階網路通訊搭建 socket 連線
- 透過遠端方法調用（RMI）將應用程式分散到多個系統上

程式語言中，Java 首創以“一個”強大的「跨平台網路程式庫」(cross-platform network library）來提供以上這些網路功能。本書將會展現這個程式庫的威力與精緻之處。本書的目的是讓你能夠將 Java 當成正式的網路程式設計平台。為此，本書不僅會提供網路的一般背景知識，也會詳細討論 Java 對網路程式設計的各項支援措施。你將會學習到如何寫出能夠透過 Internet 共享資料的 Java 程式，來實現遊戲、協同作業、軟體更新、檔案傳輸等等應用。你還會學到 HTTP、SMTP、TCP/IP 的運作原理，以及支援 Internet 與 Web 的其他協定。讀完本書之後，你所獲得的知識與工具，足以讓你寫出能完全發揮 Internet 潛力的新一代軟體。

關於第三版

1996 年，我在本書第一版的第一章中，大談 Java 的動態分散式應用。在本書後續的版本中，很高興能夠看到當初的預言一一兌現。程式設計人員現在可以用 Java 來查詢資料庫伺服器、監測網頁、控制望遠鏡、管理多人遊戲等等，這些全都是靠 Java 的 Internet 存取能力做出來的。Java 程式設計（尤其是網路程式設計）已經脫離推廣階段，而進入實務領域，可以實際發揮用途。並非所有網路軟體都是用 Java 寫成的，但是這方面的努力從不間斷過。由 C 所寫成的網路用戶端 / 伺服端基礎措施，已陸續被純 Java 所取代。現在為較新的協定，像是 Gnutella 和 Freenet，撰寫用戶端時都會優先選擇 Java。雖然在可見的未來，Java 還不致於全面取代 C 在網路程式設計方面的地位，但已經有愈來愈多人願意使用以 Java 撰寫而成的網頁瀏覽器、網頁伺服器以及其他軟體。

本書第三版有一章全新的內容，用來描述自 Java 1.1 推出 reader 和 writer 之後，在網路程式設計方面有哪些極重要的發展。我指的當然就是 `java.nio` 套件中新的 I/O API。它能夠進行異步、非阻塞式 I/O 操作，此能力對需要高效能的網路應用程式（尤其是伺服器）來說，極為重要。這消除了以 Java 撰寫網路伺服器最後的障礙。許多其他章節也做了相應更新，以便反映和利用這些新的 I/O API。

第三版也涵蓋了 Java 1.4 和 1.5 中 `java.net` 和支援套件裡許多其他的小變動和更新。此處所提到的新類別包括 `CookieHandler`、`SocketAddress`、`Proxy`、`NetworkInterface` 和 `URI`。IPv6 已經成為事實，所以此處也會做全面的介紹。Java 最新的兩個發行版本為既有的類別增添了許多其他的方法，這些都會在相關的章節進行討論。我還對本書進行了大幅的改寫，以反映 Java 程式設計（尤其是網路程式設計）的實際變動。第 3 版不再強調 Applet 和 CGI 程式，你將會發現它們的內容已經被更

換成，遠端程式碼執行（remote code execution）和伺服端環境（server-side environment）的更一般性的討論，而不管它們是如何實作的。

事實上，我花在改寫這本書的時間，相當於我當初撰寫第一版的時間。先前曾說過，本書有五篇全新章節，而其它十四篇保留下來的章節幾乎是徹底改寫，除了將內容更新到符合最新的發展現況，更大幅擴充了相關題材。老實說，就算不理會所有範例的改變，改版後的這本書也比第一版要好多了。我衷心期望本書是比第一版更優秀、更經典、更精確、也更有樂趣的網路程式設計教材。

當然，第三版的內文經過重新的整理。儘管僅有一章的內容是全新的，但是原有的 18 章也都經過大幅的改寫與擴充，好讓本書符合最新的發展，以及讓本書更加清晰、更具吸引力。希望讀者將會發現，做為一本 Java 網路程式設計的指南和參考書，第三版比第二版更充分、更更長久、更準確以及更有趣。

本書的編排

本書前三章將會概要介紹網路和網路程式的運作原理。第 1 章《為什麼用 Java 寫網路程式？》將會介紹 Java 的網路程式設計，以及可能的應用方向，探討有哪些獨特的應用非得靠 Java 和網路搭配在一起才可能實現，相信本章應該能引起所有讀者的興趣。第 2 章《網路的基本概念》和第 3 章《Web 的基本概念》將會解釋 Internet 與 web 的運作細節，而這些都是程式設計人員必備的基本知識。第 2 章將會說明 Internet 賴以運作的協定，像是 TCP/IP 與 UDP/IP 等等。第 3 章將會說明讓網站得以運作的標準，像是 HTTP、HTML 和 REST。如果你在其他平台上有其他語言的網路程式設計經驗，應該可以略過這兩章不讀。

接下來的兩章，會將焦點移到 Java 程式設計的兩項關鍵機制：I/O 與 threading。幾乎所有網路程式都需要用到這兩項機制，但是它們卻時常遭到誤解，甚至被誤用。第 4 章《串流》將會探討 Java 的典型 I/O 模型，儘管已經有新的 I/O API，但是這些模型不會很快就被棄之不用，在大多數的用戶端應用程式中，它們仍然是用來處理輸入和輸出的首選。我們必須先瞭解 Java 在一般情況下如何處理 I/O 的，才有辦法瞭解 Java 是如何處理網路 I/O 這種特殊情況的。第 5 章《執行續》將會探討「多緒」（multithreading）與「同步化」（synchronization）的議題，並特別著重在如何將它們應用在網路伺服器與異步 I/O 上。有經驗的 Java 程式設計人員應該可以大略翻閱或直接跳過這兩章。然而，第 6 章《查詢 Internet 位址》則是任何人都應該要讀的，本章將會示範 Java 程式如何透過 `InetAddress` 類別與網域名稱系統互動，基本上幾乎所有的網路程式都需要用到 `InetAddress` 類別。讀完第六章之後，你可以直接研讀你感興趣的其他主題，不過特定的章節之間存在著若干關連性。圖 P-1 顯示了你可以採用哪些可能的途徑來閱讀本書。

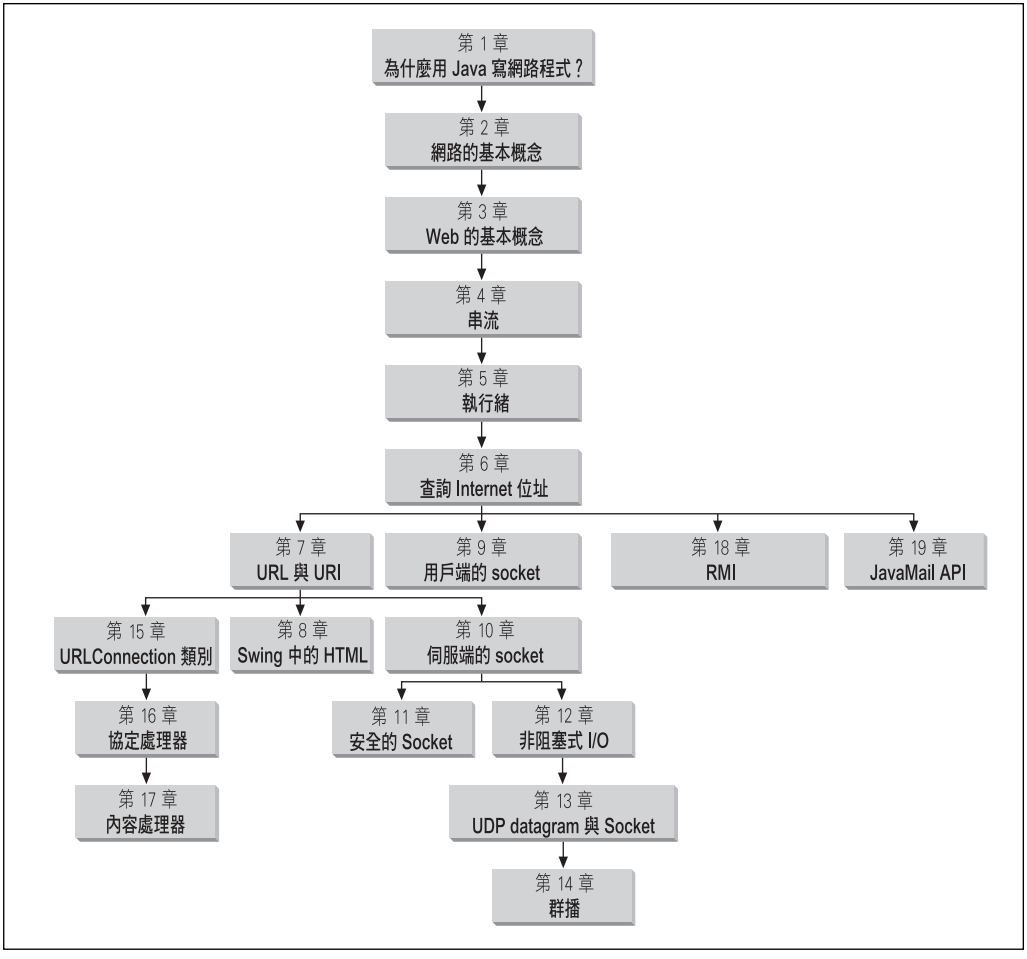


圖 P-1：本書各章的關連性。

第 7 章《URL 與 URI》將會探討 Java 的 **URL** 類別，此類別是一個可以從多種伺服器取得檔案與資訊的通用工具。**URL** 類別讓你得以不必理會協定細節，就能從網路伺服器下載檔案與文件。也就是說，用來連接 HTTP 伺服器或是在自己的硬碟上讀取檔案的程式碼，也同樣可以用來連接 FTP 伺服器。

一旦能夠從伺服器取得 HTML 檔案，下一步就是對檔案做些事。HTML 的「剖析」(pharsing) 與「呈現」(rendering)，是網路程式設計人員所面臨的最困難挑戰之一。第 8 章《Swing 中的 HTML》將會介紹一些鮮為人知但可用來剖析、呈現 HTML 檔案的類別，它可讓你卸下自己處理 HTML 語法的重擔，而將責任推給 Sun 公司。

第 9 章到第 11 章將會探討 Java 用來存取網路的低階 socket 類別。第 9 章《用戶端的 socket》將會介紹 Java socket API，尤其是 `Socket` 類別。它還會示範如何寫出能與各種 TCP 伺服器（包括 whois、finger 和 HTTP）互動的網路用戶端程式。第 10 章《伺服端的 socket》將會示範在 Java 中如何使用 `ServerSocket` 類別寫出各種協定的伺服器。第 11 章《安全的 Socket》將會示範如何使用 Secure Socket Layer（SSL）與 Java Secure Sockets Extension（JSSE）來保護用戶端和伺服端之間的通訊。

第 12 章《非阻塞式 I/O》涵蓋了 Java 1.4 所引進的新 I/O API。這些 API 是專為網路伺服器而設計的。它們能夠讓程式在試圖讀或寫 socket 之前，得知連線是否已準備就緒。這讓單一執行續得以同時管理多個不同的連線，從而讓虛擬機（virtual machine）的負荷大為減少。對於不需要同時開啟多個連線的小型伺服器或用戶端來說，這些新的 I/O API 用處不大；但是對於那些高吞吐量的伺服器來說，新的 I/O API 卻能大幅提升伺服器的效能，使得它們能夠以網路所能處理的速度，儘可能傳送最多的資料。

第 13 章《UDP datagram 與 Socket》將會介紹 User Datagram Protocol（UDP）協定與相關的 `DatagramPacket` 和 `DatagramSocket` 類別，並示範如何透過它們來進行快速但不可靠的通訊。最後，第 14 章《群播》將會示範如何使用 UDP 同時與多部主機通訊。Java 用來存取網路的所有其他類別，全都是靠以上這五章所介紹的基本類別來達成的。

第 15 到第 17 章則會深入探討 `URL` 類別底層的基礎架構，以及介紹 Java 特有的概念，協定處理器（protocol handler）與內容處理器（content handler），這兩項利器讓你得以寫出可動態擴充的軟體，自動學習新的協定與媒體型態。第 15 章《URLConnection 類別》將會說明做為 `URL` 類別之引擎的 `URLConnection` 類別，示範如何發揮該類別之 public API 的優點。第 16 章《協定處理器》的焦點也是擺在 `URLConnection` 類別，不過這一次是從另一個角度來看；本章將會示範如何藉由繼承 `URLConnection` 類別來為新的協定和 URL 建立處理器。最後，第 17 章《內容處理器》將會探索 Java 支援新媒體類型但似乎即將取消的機制。

第 18 與 19 章將會探討 Java 網路程式特有的兩套較高階的程式介面：遠端方法調用（Remote Method Invocation，簡稱 RMI）和 JavaMail API。第 18 章《RMI》將會介紹遠端方法調用這個強大的機制，以及說明如何利用此機制來撰寫分散式 Java 應用程式，讓它不僅能夠跨異質系統執行，而且能夠像非分散式程式那樣，以直接的方法調用來進行通訊。第 19 章《JavaMail API》將會介紹 JavaMail 這個標準的 Java 擴充；JavaMail API 是「用來與 SMTP、POP、IMAP 及其他電子郵件伺服器通訊之低階 socket」的替代方案。這兩種 API 都可以為分散式應用程式提供低階協定的高階替代方案。

讀者的定位

本書會假定你已經對 Java 語言與其開發環境有基本的認識，並且具備一般的物件導向程式設計概念。本書並不打算成為 Java 語言的基礎教材。你應該熟悉 Java 語言的語法，而且至少曾經寫過簡單的程式與 applet。此外，你還應該熟悉基本的 AWT 和 Swing 程式設計。書中也會涵蓋一些需要更深入瞭解的主題 — 像是「執行續」和「串流」— 而且至少會有簡單的介紹。

你還應該是一個熟練的 Internet 使用者。我將會假定你已經了解如何 FTP 檔案以及瀏覽網站。你應該知道 URL 是什麼以及如何找到它。儘管本書並不要求你是一個超級的網站設計人員，不過你至少應該知道如何撰寫簡單的 HTML，以及能夠發行內含 Java applet 的主頁。

然而，本書並不會假定你之前有過網路程式設計的經驗。所以，在這本書中，你應該會找到網路概念和網路應用程式開發的完整介紹。我並不會假定你能夠隨口說出若干網路術語（像是 TCP、UDP、SMTP，等等）。你將會從本書學習到瞭解這些術語的必備知識。你當然可以先把本書當成是以 socket 介面進行網路程式設計的入門書，然後在去學習 WSA（Windows Socket Architecture），好讓你能夠瞭解如何以 C++ 來撰寫網路應用程式。但是，我並不清楚你閱讀本書的實際目的是什麼：正如我稍早所說，Java 可讓你輕易寫出精妙的應用程式。

Java 的版本

Java 1.0 發行之後，Java 的網路類別幾乎已經定型，其演變速度比 core API 裡的其他成員慢得多，相較於 AWT 或 I/O，可說是幾乎沒有改變，而且只有小幅擴充。當然，幾乎所有網路程式都會大量使用 I/O 類別，甚至大量使用 GUI。在編寫本書時，我們假定你和你的客戶都使用 Java 1.1 以上版本。本書將會假設你與你的客戶至少使用的是 Java 1.1。也就是說，我會直接使用 Java 1.1 的功能 — 像是 reader 與 writer 以及新的事件模型 — 而不會進一步解釋它們的來龍去脈。

有鑑於此，— 除非存在 1.1 版的安全替代方案。對於那些 Java 1.2 之後才有的方法或類別，我會加上這樣的註解：

Java 2（即 Java 1.2）做了比較多的擴充。雖然全書幾乎是基於 Java 2 寫成的，而且 Java 2 多年以前就已經可以使用在大多數平台上，但是還不存在可用在 Mac OS 9 的 Java 2 執行和開發環境。幾乎可以確定的是，Apple 和 Sun 都不會將 Java 2 的任何版本移植到 Mac OS 9.x 或更早版本上，這使得目前安裝 Mac 的平台中，有 60% 無法進行 Java 的開發。對於號稱『write once, run anywhere』的 Java 語言來說，這可不是一件好事。此外，Microsoft 的 Java 虛擬機僅支援 Java 1.1，而且在可見的未來似乎不

可能有任何進展。因此，儘管我不避諱使用 Java 2 才有的一些方便的功能（像是 `InputStreamReader` 的 ASCII 編碼功能以及 *keytool* 程式），不過我會特別說明，讓你知道我的做法並非“到處可用”。當存在安全的 1.1 替代方案時，我也會特別註明。當特定的方法或類別是 Java 1.2 或之後版本的新功能時，我會在它的宣告之後加上註解，像這樣：

```
public void setTimeToLive(int ttl) throws IOException // Java 1.2
```

更麻煩的是，Java 2 有多個版本。完成本書當時，現行的發行版本是“Java™ 2 SDK, Standard Edition, v 1.4.2_05”。至少它當時就叫這個名字。Sun 可能是根據市場顧問的意見修改了名字。之前的版本中，它的名字則是眾所周知的 JDK。Sun 還使用了“Java™ 2 Platform, Enterprise Edition（J2EE™）”和“Java™ 2 Platform, Micro Edition（J2ME™）”等名字。Enterprise Edition（企業版）是 Standard Edition（標準版）的超集，添加了許多功能，像是可以為分散式應用程式提供高階 API 的 Java Naming and Directory Interface 和 JavaMail API。這些新增的 API 多半還能用來擴充標準版，本書就是如此。Micro Edition（小型版）則是 Standard Edition（標準版）的子集，主要應用在手機、機上盒，以及記憶體、CPU 和顯示功能有限的裝置。它去除了很多程式設計人員已經了解之與 Java 相關的 GUI API，不過它卻保留了許多本書所提到的基本網路功能和 I/O 類別。最後，當本書大約完成一半時，Sun 發行了“Java™ 2 SDK, Standard Edition, v1.5”的 beta 版。它為網路 API 添加了若干東西，不過大多數既有的 API 都沒有改變。接下來的幾個月，Sun 又發行了 JDK 1.5 的另外幾個 beta 版。本書最後的內容以及所有程式碼都已經使用 JDK 1.5 beta 2 測試過。這樣即使在 JDK 1.5 正式發行之後，使用本書的內容也不會遇到任何困難。幸運的是，最後的規格和我在此處所做的討論，其間的差異微乎其微。

老實說，處理不同的版本時，最煩人的問題不是進行必要的改寫，而是要在文中標出它們。我並不想，每次在指出 Java 最新版中的新功能時，都寫上“Java™ 2 SDK, Standard Edition, v1.3”或甚至是“Java 2 1.3”。通常我只會提到 Java 1.1、Java 1.2、Java 1.3、Java 1.4 和 Java 1.5。但總地來講，網路 API 似乎相當穩定。從 Java 1.1 到 Java 1.3 都非常相似，在 Java 1.4 和 1.5 中只有若干重要的擴充。Java 1.0 之前的網路 API 只有非常少數被廢棄不用。

關於範例程式

本書所探討的方法和類別，至少都會舉一個完整的可執行範例，儘管這個範例可能很簡單。以我的經驗來看，要示範某方法的正確用法，一個可執行的範例程式是絕對必要的。如果沒有實用的範例，很容易會陷入一堆術語中，或是錯過作者可能自己也沒搞清楚的要點。Java API 的說明文件就是個典型例子：你可以看到每個方法與類別的冗長說

明，但是很少有範例，也沒交代清楚應用時機與相關用途。在本書中，我會避免這種寫作心態，寧可用過多的範例來反覆說明，也不願意漏掉一個可供參考的範例，使你花太多時間在理解我想說明的主題上。如果你對某個知識點很清楚，就可以跳過有關的說明。當然，如果讀者都已經瞭解我想說明的主題，大可跳過這些部分。你並不需要親自鍵入和執行書中每一個範例程式，但如果有某個方法讓你感到困惑，我保證你在書中至少會看到一個可用來測試該方法的範例。

每一章都至少會有一個（通常會有好幾個）較複雜的範例程式，用來示範該章所介紹之類別和方法的實際用途。這些複雜的範例常會用到本書並未討論到的 Java 功能。事實上，在許多程式的原始程式碼中，網路元件只佔其中的一小部分，通常是最不困難的部分。然而，倘若不使用 Java 這種以網路為中心的語言，情況肯定會完全改觀。看似簡單的網路程式碼，反映出網路功能已經成為 Java 的核心功能，而非應用程式本身要操心的瑣事。本書的所有範例可直接從 *http://www.cafeaulait.org/books/jnp3/* 下載，通常這些範例都會有所修正和擴增。

本書會假定你使用的是 Sun 的 Java Development Kit。我已經在 Linux 上測試過所有範例，以及在 Windows 和 Mac OS X 上測試過其中許多範例。照理說，只要你用的編譯器和虛擬機器支援 Java 1.2，所有範例都應該能順利過關，就算只用 Java 1.1，大多數範例也應該可以過關。當範例程式需要用到 Java 1.3、1.4 或 1.5 時，我也都會很清楚地加以註明。

本書慣例

內文採用細明體，如你現在所見。

粗體字用於：

* 第一次出現的術語。

定寬字用於：

* 完整或片段的程式碼範例
* 出現在 Java 程式碼中的關鍵字、算符、資料型別、方法名稱、類別名稱以及介面名稱
* 程式的輸出
* 出現在 HTML 文件中的標籤

定寬粗體字用於：

* 應該動手鍵入的命令列和選項

斜體字用於：

* 路徑名稱、檔案名稱以及程式名稱（但是，如果程式名稱也是 Java 類別的名稱，則會使用定寬字，就像其他類別那樣）
* 主機和網域名稱（*java.oreilly.com*）
* URL（*http://www.cafeaulait.org/slides/*）

在內文的段落之前，時常會看到片段的程式碼，像這樣：

```
Socket s = new Socket("java.oreilly.com", 80);
if (!s.getTcpNoDelay(  )) s.setTcpNoDelay(true);
```

看到這種不完整的程式片段時，意味著，還必須使用適當的 `import` 敘述才行。就本例而言，你需要假定 `java.net.Socket` 已經被匯入。

有些範例同時包含了使用者的輸入與程式的輸出。為了有所區別，使用者的輸入將會以定寬粗體字呈現，例如下面這個來自第 9 章的範例：

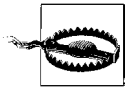
```
% telnet rama.poly.edu 7
Trying 128.238.10.212...
Connected to rama.poly.edu.
Escape character is '^]'.
This is a test
This is a test
This is another test
This is another test
9876543210
9876543210
^]
telnet> close
Connection closed.
```

Java 程式語言是大小寫有別的（case-sensitive）。也就是說，`Java.net.socket` 和 `java.net.Socket` 是兩回事。會區分大小寫的程式語言，有時會使得作者的構句與標準的英文文法不符。一般而言，要改寫句子，使其兼顧技術描述和英文文法的正確性，並非完全不可能，而我也會盡力這麼做。然而，有些狀況實在無法兼顧，此時我只好選擇讓標準的英文文法成為輸家。為了保持這項原則，當我在內文提到某類別（class）或是某類別的某個體（instance）時，我會通常會採用首字母大寫的方式，例如 `ServerSocket`。

在本書中，我會使用英式的標點符號慣例，也就是說，逗號和句號會被放在雙引號之外，而不像美式的做法，將逗號放在雙引號之內。雖然我學的是美式文法，但是我認為英國系統比較符合邏輯，即使我明知道在程式原始碼中，將句號、逗號、分號放在雙引號外，與它們放在雙引號內的意義是完全不一樣的。

最後，雖然本書有許多範例沒有重複利用的價值，但是我所開發的類別中仍有少數具有實用價值。你可以放心地將我的程式碼運用在你的程式中，不必經過任何特殊的許可，因為我願意將這些程式碼貢獻出來，當成**公共財（public domain）**。要特別聲明的是，解釋這些程式碼的文字並非公共財。我依照 `java` 套件的分類慣例，將那些有利用價值的類別集中放在 `com.macfaq` 套件中。以第四章的 `SafePrintWriter` 類別為例，我將它收錄在 `com.macfaq.io` 套件中。使用這些類別時，別忘了，編譯好的 *.class* 檔案必須放在與套件結構相符的目錄之下（該目錄的“邏輯結構頂層”必須在類別路徑中），而且你必須在使用它們之前，在你自己的類別中匯入（`import`）它們。本書的網頁（*http://www.cafeaulait.org/books/jnp3/*）可以找到一個 *jar* 檔，本書所有可重複利用的工具類別都包裝在此檔案中，你可將此 jar 檔放在你的類別路徑中。

用於提示、建議或一般注意事項。



用於警告或告誡。

批評指教

我喜歡聆聽讀者的意見，不管是如何讓本書能夠更好的一般建議、指正、其他你認為應該納入本書的議題，或是你個人在網路程式設計方面的經驗甘苦談。你可以用電子郵件與我聯絡（*elharo@metalab.unc.edu*）。不過，請務必諒解我每天會收到好幾百封電子郵件，可能無法親自一一回覆。如果你來信時有註明你是本書的讀者，將會比較有機會得到我的親自回信。如果你發覺某程式的效果不如你的預期，請嘗試將程式簡化到最單純的狀況，使其能重現你所發現的瑕疵，然後將整個程式貼到電子郵件的「內文」；來源不明的夾帶檔案會被我的濾信系統刪除。最後，我要拜託各位，務必使用你想收到回信的電子郵件帳號來寄信，並確定你的 Reply-to 所指定電子郵件地址是正確的。試想，如果你花了好幾個小時細心研究一個使你感興趣的問題，並詳細寫下研究所得的答案，卻在事後才發覺無法回信到正確地址，那會是多麼令人沮喪的一件事。

我還是那句老話「如果你喜歡這本書，請告訴你的朋友。如果不喜歡，請讓我知道。」我會特別注重讀者所提出的指正。雖然這已經是我的第八本書了，但我還沒出版過完美的作品，不過我會繼續朝此方向努力。儘管 O'Reilly 的編輯群與我都下了不少心力，但是我相信本書必定仍有疏漏或錯字，甚至是令我難堪的嚴重錯誤，只是我事先不知道它們在哪裡而已。如果讀者發現了這樣的錯誤，請務必讓我知道，我會將已發現的錯誤

公佈在本書專屬網頁（*http://www.cafeaulait.org/books/jnp3/*）以及 O'Reilly 的網頁（*http://www.oreilly.com/catalog/javanetwk/errata/*）。在你回報錯誤之前，也請先到這兩個網頁看看，是否該問題已經有人發現並修正好了。

建議與問題

歐萊禮公司是世界性的電腦資訊出版公司。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

網頁：*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）

與本書有關的線上資訊（包括勘誤、範例程式、相關連結）：

原文書

http://www.oreilly.com/catalog/javanp3/

中文書

http://www.oreilly.com.tw/product_java.php?id=a177

感謝

這本書是眾人努力的結果，我的編輯：Mike Loukides，使得這本書得以問世，他在我寫作的過程中提供了許多有用的建議。Peter Parnes（筆名：Peppar）博士協助我完成《群播》那一章。技術編輯群也功不可沒，他們糾正了本書不少的錯誤與疏漏之處。Simon St. Laurent 協助我勾勒出本書各章所應該涵括的議題。Scott Oaks 以他在“執行續”方面的專長協助我，糾正我在第五章所犯的一些錯，其中甚至包括具多緒程式設計經驗的老手都可能會不經意犯下的錯誤。Ron Hitchens 對於新的 I/O API 為我考慮到了許多未被注意的方面。Marc Loy 和 Jim Elliott 審閱了書中一些最具風險的素材。

Jim Farley 和 William Grosso 在遠端方法調用方面提供了我許多有用的意見和援助。Timothy F. Rohaly 致力於督促我保持優良的程式設計風格，確定我關閉了每一個曾經開啟的 socket、並攔截所有可能發生的異常、寫出最乾淨、最安全、最可能當成典範的程式碼。John Zukowski 找出了我大量的疏失，讓我不致於出書之後才出糗。讓我感到不可思議的是 Avner Gelb 他那銳利的眼光，他指出許多我和其他編輯（甚至成千上萬的第一版讀者）都沒注意到的瑕疵。

我也要感謝本書的發行人 — Tim O'Reilly — 但不是基於「慣例」而感謝他，而是他的確為出版公司、作者、編輯群、製作群樹立了一致的格調。我尤其認為 Tim O'Reilly 理應得到另一項特殊功勞：他使得 O'Reilly Media 成為作者最佳的寫作場所之一。如果說本書若少了哪個人就寫不成的話，這個人就是 Tim。如果你（讀者）發現 O'Reilly 的書總是比市面上其他競爭者的書更值得一讀，真的應該要歸功於 Tim 對品質的堅持。

感謝我的經紀人，David Rogelberg，讓我能遠離辦公室專心致力於本書的寫作。感謝 *ibiblio.org* 的全體工作人員，最近這幾年協助我能以各種方式與讀者做更好的溝通。曾經在第一版和第二版對我提出建言的每一位讀者，也都是促使我改版的推動者。以上所提及的每一個人，都是我要感謝的對象。最後，照例，我要把最大的感謝保留給我老婆，Beth，沒有她的支持與協助，本書將不可能存在。

—Elliotte Rusty Harold
elharo@metalab.unc.edu
2004 年 9 月 22 日