# 前言

歡迎閱讀《*Continuous API Management*》第二版。本書的第一版在 2019 年出版，它的第一段是這樣寫的：

> 隨著社會與商業日益數位化，大眾對互相連結的軟體的需求已呈爆炸式成長。由於應用程式開發介面（*API*）促進軟體的相連，*API* 已經成為現代組織最重要的資源了。但是有效地管理這些 *API* 是一項新的挑戰。若要充分發揮 *API* 的價值，你必須知道如何管理它們的設計、開發、部署、成長、品質與安全，以及處理關於背景、時間與規模的複雜因素。

近年來，關於 API 管理的成長與挑戰並沒有什麼變化。好消息是，自第一版以來，更多工具、訓練、經驗已協助了 API 管理領域的成長與成熟。不太好的消息是，作者看到很多組織還在辛苦地使用 API 來滿足連結大眾、服務與公司的需求。新版讓我們有機會提供公司的最新進展、分享一些新的成功故事，以及完善我們在 2018 年介紹的一些教材。

雖然我們加入新例子，並且更新了既有的例子，但是在新版裡，我們仍然保留相同的基本方法和大綱。希望這些改變可以幫助你在 API 持續管理的道路上延伸你自己的旅程。

## 誰該閱讀這本書？

如果你準備開始編寫 API 程式，並且想要了解將來的工作，或是你已經有一些 API 了，但是想要知道如何管理它們，這本書就很適合你。

本書試著建構一個可在各種背景之下運用的 API 管理框架，教你如何管理一個供全世界的開發者使用的 API，以及如何在專為內部開發者設計的微服務架構內，建立複雜的 API 組合，和介於兩者之間的所有知識。

本書盡量保持技術中立，我們提供的建議與分析適用於任何 API 結構，包括 HTTP、CRUD、REST、GraphQL 與事件驅動型互動。本書適合希望改善 API 決策的任何人。

# 本書的內容

本書包含筆者多年來設計、開發、改善自己和別人的 API 所累積的知識，我們將所有經驗都濃縮在這本書裡面了。我們發現，高效的 API 開發有兩個核心要素：從產品的角度看事情，以及建立正確的團隊。我們也發現，管理這項工作有三個基本要素：治理、產品成熟度與園林設計。

這五項 API 管理元素是建構成功的 API 管理專案的基礎。本書將介紹這些主題，並教導你如何將它們融入你自己的組織背景。

## 大綱

本書的章節經過刻意安排，希望隨著章節的進展而增加管理問題的範圍。我們會先介紹「決策導向治理」和「API 即產品」的基本概念，然後介紹建構 API 產品時必須管理的所有工作。

我們將從單一 API 的觀點開始探討「變動 API」是什麼意思，接下來，當我們深入研究「變動 API」的意義和「API 的成熟度如何影響這些變動決策」時，我們將加入時間方面的問題。接著，我們會討論進行變動的團隊與人員。在本書的後半部，我們要討論規模的複雜度，以及管理 API 產品園林的挑戰。

以下是各章的摘要：

- 第 1 章「管理 API 的挑戰與承諾」將介紹 API 管理領域，並解釋為何有效管理 API 如此困難。
- 第 2 章「API 治理」從決策導向工作的角度來討論治理，它是 API 管理的基本概念。
- 第 3 章「API 即產品」建立 API 即產品的觀點，以及說明為何它是任何一次 API 決策的重要元素。
- 第 4 章「API 產品的十大支柱」概述 API 產品領域的十大基本工作支柱。這些支柱構成了一套必須管理的決策任務。
- 第 5 章「持續改善 API」深入介紹持續變動 API 是什麼意思。本章介紹「持續變動心態」的必要性，並說明你將遇到的各種 API 變動及其影響。

- 第 6 章「API 風格」是這一版加入的新章節。本章介紹我們造訪世界各地的公司時看到的五種最常見的 API 風格，並探究各種風格的優缺點，以協助你在遇到各種用例（use case）時選擇最合適的風格。
- 第 7 章「API 產品週期」介紹 API 產品生命週期，這個框架可以協助你在 API 產品的整個生命週期中，管理橫跨十大支柱的 API 工作。
- 第 8 章「API 團隊」探討 API 管理系統的人員元素，本章將探討 API 產品生命週期中，API 團隊的典型角色、責任與設計模式。
- 第 9 章「API 園林」在 API 管理問題中加入規模的觀點，介紹同時變動多個 API 時需要處理的八個 V—— 多樣性、詞彙、數量、速度、脆弱性、能見度、版本管理與波動性。
- 第 10 章「API 園林之旅」介紹一種持續性園林設計法，可大規模地、持續地管理 API 的變動。
- 第 11 章「在持續演變的園林中管理 API 週期」會從園林的角度看待 API 即產品，說明當園林圍繞著它而演變時，API 工作會怎麼改變。
- 第 12 章「繼續這趟旅程」將已介紹的 API 管理故事串聯起來，並建議你如何為將來做好準備，以及如何立即開始你的旅程。

## 本書沒有談到的東西

API 管理領域十分廣大，且環境、平台與協定有許多差異。受限於本書的著作時間與空間，我們無法介紹 API 工作的所有具體實作方法。本書不是設計 REST API 或挑選安全閘道產品的指南。如果你想要了解如何編寫 API 程式，或設計 HTTP API，那麼這本書不適合你。

雖然本書有些範例討論特定的實作，但這不是一本專門討論 API 實作的書（好消息是有大量的書籍、部落格與影片可以滿足你的需求）。取而代之的，本書處理的是一個很少人處理的問題：如何在一個複雜的、持續變動的組織系統中有效地管理 API 建構工作。

## 本書編排方式

本書使用以下的編排規則：

第一章

# 管理 API 的挑戰與承諾

管理是融合藝術、科學與工藝的行動。

—Henry Mintzberg

根據 2019 年的 IDC 報告，有 75% 的受訪公司預計在未來十年內進行「數位轉型」，而且預計有 90% 的新 APP 將以 API 支持微服務架構[1]。也有報導指出，對注重 API 的組織來說，有高達 30% 的收入是透過數位管道產生的。與此同時，這些公司認為採用 API 的主要障礙是「複雜度」、「安全性」與「治理」。

最後，這是最關鍵的結論之一：「若要定義正確的 APP 架構，你必須深入了解治理、管理和編配這些基本技術組件的相關挑戰。[2]」這份研究如同我們在本書的第一版引用的 Coleman Parkes 所做的研究（*https://oreil.ly/pAWGs*），既包含鼓勵，也提出告誡。

在過去幾年來，我們發現一個有趣的趨勢：「擁有 API」與「沒有 API」之間的差距越來越大。例如，對於「你的公司有 API 管理平台嗎」這個問題，72% 的媒體和服務公司回答「有」，但是在製造業裡，只有 46% 的公司回覆肯定的答案[3]。所有跡象表明，API 將會繼續推動業務的成長，任何經濟領域的公司都必須緊鑼密鼓地迎接數位轉型的挑戰。

好消息是，已經有很多公司成功地管理它們的 API 專案了，但壞消息是，他們的經驗與專業知識不太容易分享，或無法普遍獲得。這有幾個原因。一般來說，擅長管理 API 的組織單純是因為太忙了，沒有時間分享他們的經驗。少數情況下，根據我們的訪談，有些公司對分享 API 管理專業知識非常謹慎，他們十分確定 API 技術是一種競爭優勢，所以慢條

1 Jennifer Thomson and George Mironescu, "APIs: The Determining Agents Between Success or Failure of Digital Business," IDC, *https://oreil.ly/9yshw*.

2 Thomson and Mironescu, "APIs: The Determining Agents Between Success or Failure of Digital Business."

3 Ibid.

斯理地公開他們發現的技術。最後，雖然有些公司會在公開的會議上，或透過文章與部落格文章分享經驗，但他們分享的資訊通常是針對特定公司的，很難轉化成廣泛組織的 API 專案。

本書試著解決上述的最後一個問題：將特定公司的典範轉化適合所有組織的共享經驗。為此，我們訪問了數十家公司，採訪了許多 API 技術專家，試著在這些公司和我們分享，以及和大眾分享的案例中找出共同點。本書有幾個貫穿全書的主題，我們將在這個介紹性章節中分享它們。

首先，你必須確定別人口中的 API 對他們而言是什麼意思。「API」可能只是代表介面（例如 HTTP 請求 URL 與 JSON 回應），也可能代表將一項可造訪的服務投入生產環境所需的程式碼與部署元素（例如 `customerOnBoarding` API），此外，我們有時會用「API」來代表一個運行中的 API 實例（例如，在 AWS 雲端運行的 `customerOnBoarding` API vs. 在 Azure 雲端運行的 `customerOnBoarding` API）。

在管理 API 時，另一項重要的挑戰是區分設計、建構與發表單一 *API* vs. 支援和管理多個 *API*（我們稱之為 *API* 園林（*landscape*）[譯註]）之間的差異，我們將用大量的篇幅來討論這個光譜的兩個極端。處理單一 API 所帶來的挑戰的案例包括 *API* 即產品（*API-as-a-Product*，AaaP）等概念，以及建立、維護 API 所需的技術（我們稱之為 *API* 支柱（*pillar*））。我們也會討論 API 成熟度模型的作用，以及如何隨著時間的變遷而處理演變，這是非常重要的 API 管理層面。

在光譜的另一端是 API 園林管理。你的園林，就是來自所有商務領域、在所有平台上運行、被你公司的所有 API 團隊管理的 API 組合。園林帶來的挑戰有許多層面，包括規模與範圍將如何改變 API 的設計與實作方式，以及大型的生態系統為何僅僅因為規模擴大就會增加它們的波動性與脆弱性。

最後，我們要討論 API 生態系統的管理決策程序。根據我們的經驗，這是為你的 API 專案制定成功的治理計畫的關鍵。事實上，你的決策方式必須隨著你的園林而改變，堅守舊治理模式可能限制 API 專案的成功，甚至為既有的 API 帶來更多風險。

在採討如何處理這兩項挑戰（單獨的 API 以及 API 園林）之前，我們先來看一下兩個重要的問題：什麼是 API 管理，以及為何它如此困難？

[譯註] landscape 本身有景觀、地貌等含意，本書用它來代表一家公司的所有 API，譯者將它譯為「園林」。

# 什麼是 API 管理？

如前所述，API 管理並非只與設計、實作與發表 API 有關。它也包含 API 生態系統的管理、如何在組織裡面發表決策，甚至將現有的 API 遷移到正在成長的 API 園林等過程。在這一節，我們要花一點時間討論每一個概念，但首先，我們必須討論 API 存在的終極理由，即 *API 商務*。

## API 商務

除了關於建立 API 與管理它們的細節之外，你一定要記住，這項工作完全是為了支援商業目標和目的。API 不僅僅關乎 JSON 或 XML、同步或非同步等技術細節，它們是將商業單位結合起來，以協助公司有效地公開重要的功能和知識的手段。API 通常是釋放組織現有價值的一種方式，例如，透過建立新的 APP、開發新的收入流，以及啟動新的業務。

這種觀點比較注重 API 用戶的需求，而不是 API 的製造人員和發表人員的需求。這種用戶導向的做法通常稱為「Jobs to Be Done」或 JTBD。JTBD 是哈佛商學院的 Clayton Christensen 提出的概念，他在《*The Innovator's Dilemma*》與《*The Innovator's Solution*》（Harvard Business Review Press）中深入探討這種做法的威力。如果你想要啟動與管理成功的 API 專案，它可以清楚地提醒你，API 的目的是為了解決商業問題。根據我們的經驗，善用 API 來解決商業問題的公司都將 API 視為用來「完成工作」的產品，與 Christensen 用 JTBD 框架來解決客戶問題是一樣的概念。

### 存取資料

API 為企業做出貢獻的另一種方式就是讓人們更容易取得重要的客戶或市場數據，那些數據可能與新客戶群的新趨勢或獨特行為有關。藉著讓這些資料可以安全且輕鬆地取得（並經過妥善地匿名和過濾），API 也許可以協助公司發現新的機會、實現新的產品 / 服務，甚至以更低的成本與更快的上市時間來啟動新舉措。

### 接觸產品

API 專案協助商業活動的另一種方式，是藉著製作一套靈活的「工具」（APIs）來建構新的解決方案，以免帶來高昂的成本。例如，如果你有一個 `OnlineSales` API 可讓重要的伙伴管理與追蹤他們的銷售活動，以及一個 `MarketingPromotions` API 可讓行銷團隊設計與追蹤產品推廣活動，或許你可以建立一個新的伙伴解決方案：`SalesAndPromotions` 追蹤 APP。

獲得創新機會

> 根據我們的經驗，許多公司的內部流程、做法和生產流水線雖然行之有效，卻沒什麼效率。那些機制已經存在很長的時間（有的多達幾十年），有時甚至沒有人記得組織何時（或為何）建立了某個流程。改變既有的流程並不容易，可能代價高昂。在公司內部建立 API 基礎設施可以釋放組織的創造力，有時可以繞過把關機制，在公司內部進行改善與提高效率。

我們將在第 3 章討論這些 AaaP 的重要層面。但首先，我們要來探討一下，如何簡單地解釋 *API* 這個術語的意思。

## 什麼是 API？

人們有時不但使用 *API* 這個術語來代表介面，也用它來代表功能，也就是介面背後的程式。例如，可能有人說：「我們必須快點發表新的 Customer API，讓別的團隊可以開始使用我們做好的新搜尋功能。」有時 API 只代表介面本身的細節。例如，你的團隊成員可能會說：「我想為現有的 SOAP 服務設計新的 JSON API，來支援客戶入門流程。」當然，這兩種意思都沒錯（而且它們都清楚地表達含義），只是有時它們會造成混淆。

為了釐清這種區別，以方便我們討論介面與功能，我們將介紹一些額外的術語：介面、實作與實例。

### 介面、實作與實例

API 是 *APPlication programming interface*（應用程式開發介面）的縮寫。我們用介面來使用在 API 「背後」運行的東西。例如，你可能有一個公開「管理用戶帳號的工作」的 API ，這個介面可讓開發者：

- 建立新帳號。
- 編輯既有的帳號資料。
- 改變帳號狀態（暫停或活躍）。

這個介面通常是以共享的協定來表達的，例如 HTTP、Message Queuing Telemetry Transport（MQTT）、Thrift、Transfer Control Protocol/Internet Protocol（TCP/IP）…等，並採用一些標準格式，例如 JSON、XML、YAML 或 HTML。

但它只是介面，你還要用其他的東西來執行用戶請求的工作，那些其他的東西就是所謂的實作。實作是提供實際功能的部分，實作通常是用 Java、C#、Ruby 或 Python…等程式語言寫成的。延續用戶帳號的例子，`UserManagement` 實作可能包含建立、加入、編輯與移除用戶等功能，這些功能可以用之前提到的介面來公開。

### 將介面與實作解耦

注意，上述的實作的功能是一組簡單的動作，它們使用建立、讀取、更新、刪除（CRUD）模式，但上述的介面有三種動作（`OnboardAccount`、`EditAccount` 與 `ChangeAccountStatus`）。這種實作與介面之間看似「不一致」的情況很常見，有時有很好的效果；它將各項服務的實作，與用來使用那項服務的介面解耦，讓你更容易隨時變動它們，而不會破壞它們。

我們的第三個術語是實例。API 實例是介面與實作的結合，很適合用來代表已被投入生產環境並開始運行的 API。我們會用一些指標來管理實例，以確保它們的健康，我們會註冊與記錄實例，讓開發者更容易尋找與使用 API 來解決實際的問題，我們也會保護實例的安全，確保只有經過授權的用戶可以執行操作，以及讀取 / 寫入完成那些操作所需的資料。

圖 1-1 是這三種元素之間的關係。在本書中，當我們說到「API」時，通常是指 API 的實例：介面與實作組合，完全可以營運。如果我們想要強調的東西只是介面或只是實作，我們就會那樣稱呼它們。

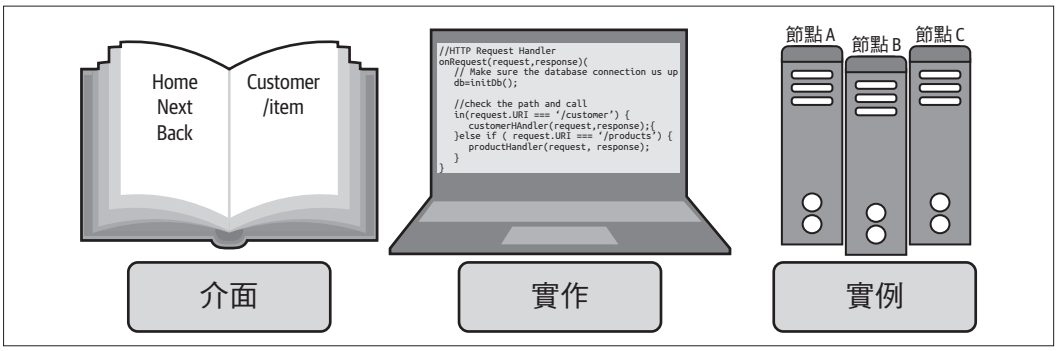


圖 1-1　三個 API 元素

## API 風格

API 的另一個重要元素是一種可以稱為風格（*style*）的東西。如同其他領域的風格（繪畫風格、裝飾、時尚、建築物），API 風格就是讓人以一致、可識別的方法來建立和使用 API 的方法。你一定要知道你的用戶端 APP 想要使用哪一種 API 風格，並在製作 API 時，提供符合該風格且一致的實作。

當今最常見的 API 風格是 REST 或 RESTful API 風格。但它只是一種可能的做法，事實上，我們在大大小小的組織內，看到越來越多人使用非 REST、非 HTTP API。事件驅動架構（EDA）的興起是這種 API 管理因素的實際案例之一。

雖然 API 有很多風格，每一種風格都有獨特的名稱，但根據我們的經驗，當你管理 API 專案時，你必須注意五種常見的風格。我們將在第 6 章討論 API 風格的重要性，並討論每一種風格。

幾乎沒有公司在整個公司內部只依靠一種風格。而且，你實作的任何一個風格都不太可能永遠不變。在設計、實作與管理 API 生態系統的時候考慮風格，是建立成功、穩定的 API 專案的關鍵因素。

多風格的 API 帶來成功管理 API 專案的另一個重要的層面：以連貫、一致的方式來治理許多 API。

## 不僅僅是 API

API 本身（介面與實作的技術細節）只是完整故事的一部分而已。在 API 的生命週期之中，設計 / 建構 / 部署等傳統元素當然也很重要。但是管理 API 通常也意味測試它們、記載它們，以及在網站入口發表它們，讓正確的受眾（內部開發者、伙伴、不知名的第三方 APP 開發者…等）可以找到它們，並學會正確地使用它們。你也要保護 API、在執行期監測它們，以及在它們的整個生命週期中維護它們（包括處理變動）。這些額外的 API 元素就是我們所謂的 *API 支柱*：它們是所有 API 都需要的元素，也是所有 API 專案經理都必須處理的元素。我們會在第 4 章深入討論支柱，在那裡，我們將探討建立與維護健康的 API 的十大關鍵方法。

關於這些實踐領域，有一個好消息是它們不限於任何單一的 API。例如，妥善記載 API 的技能可以從一個 API 團隊傳授給下一個 API 團隊。學習正確的測試技巧、安全防護的模式…等也是如此。這也意味著，即使你讓不同的團隊負責各個 API 領域（銷售團隊、產品團隊、後台團隊…等），不同團隊的成員之間也有「交叉」利害關係[4]。

管理 API 的另一個重要層面是管理「API 製作團隊」並授予權力。我們會在第 8 章進一步討論不同的組織如何運作它。

## API 成熟階段

API 支柱只是整件事的一部分。在專案中的每一個 API 都會經歷它自己的「生命週期」，生命週期是一系列可預測且實用的階段。知道目前處於 API 旅程的哪個階段可協助你決定此刻應投資多少時間與資源在 API 裡。了解 API 有多麼成熟可讓你認出多個 API 的相同階段，並協助你為各個階段需要付出的時間與精力做好準備並做出回應。

從表面上看，在設計、建構與發表 API 時，你應該處理所有的 API 支柱，但是現實並非如此。例如，在 API 的早期階段，通常你要把重心放在設計與建構層面上，不要花太多精力在記錄上。在其他的階段（例如，當你已經將雛型送給 beta 測試者時），比較重要的事情是花更多時間監測 API 的使用，以及防止它被濫用。了解成熟階段可協助你分配有限的資源來取得最大的效果。我們將在第 7 章帶你經歷這個過程。

## 超過單一 API

許多讀者知道，當你開始管理許多 API 時，事情會開始變調。我們有些客戶需要建構、監測和管理上千個 API，在這種情況下，我們不太在乎個別 API 的實作細節，而是更關注一件事：讓這些 API 在一個不斷成長、動態的生態系統中共存。如前所述，我們將這種生態系統稱為 API 園林，本書的後半部分會用幾個章節專門討論這個概念。

此時，你的挑戰大部分都是如何確保某種程度的一致性，同時不至於為了集中管理和審查所有的 API 細節，而造成瓶頸和速度的下降。我們通常將這些細節的職責分配給各個 API 團隊，並將中央的管理 / 治理重心放在規範 API 之間的互動方式上，確保有套核心的共享服務或基礎設施（資安、監測…等），讓所有的 API 團隊使用，通常也會指引和輔導更多的自主團隊。也就是說，我們通常必須擺脫常見的集中指揮和控制模式。

4 音樂串流服務 Spotify 將這種交叉群體稱為 *guilds*。若要進一步了解這個主題，見第 188 頁的「擴大你的團隊」。

當你在組織中更深入地分配決策權和自主權時，你將面臨一項挑戰：組織的高層往往無法看到團隊層面的重要活動。在過去，團隊可能需要徵求許可，才能採取某項行動。將額外的自主權授予各個團隊的公司會鼓勵他們採取行動，不需要經過上層的審查與許可。

在管理 API 園林時出現的挑戰通常與規模和範圍有關。事實上，隨著 API 專案的增長，它不但變得更大，它的形態也會變化。這是接下來要討論的主題。

# 為何 API 管理很難？

我們在本章開頭說過，雖然大多數公司都已經啟動 API 專案了，但有些經濟領域的 API 比其他領域的更先進。為何如此？為何有些公司的 API 比其他公司更好？常見的挑戰有哪些？如何協助你的公司克服它們？

當我們拜訪世界各地的公司，討論如何管理 API 生命週期時，有一些基本的主題開始浮現出來：

範圍

當軟體架構中央團隊在治理 API 時，隨著時間的不同，他們應該將焦點放在哪裡？

規模

當專案從少數幾個小團隊擴展成全球計畫時，在研發 API 初期行之有效的做法，通常無法隨著專案擴展。

標準

隨著專案的成熟，管理與治理工作必須從「詳細地建議如何設計與實作 API」，變成較廣義的「API 園林標準化」，讓團隊可以自行做出更多且更詳細的決策。

從本質上講，持續平衡這三個元素（範圍、規模與標準）可推動健康、持續成長的 API 管理專案。因此，這些元素值得深入探討。

## 範圍

若要營運健康的 API 管理專案，有一項很大的挑戰在於拿捏中央單位的控制程度，更有挑戰性的是，隨著專案的成熟，適當地調整控制的程度。

在專案的早期，我們應該把重心放在設計 API 的細節上。當 API 處於起步階段時，這些設計細節可能直接來自製造 API 的團隊，他們可能會研究「坊間」的現有專案，為他們打算製造的 API 風格選擇適當的工具和程式庫，然後開始動手製造那個 API。

在這個 API 生命週期「第一階段」中，提供詳細的建議，以及定義明確的角色，可以帶來早期的成功。第 3 章與第 4 章會介紹我們認為對剛開始 API 之旅的公司有幫助的教材。

在這個 API 專案的「早期階段」裡，一切都是新的，所有問題都是公司第一次遇到的（與解決的）。這種最初的經驗往往被公司記錄下來，成為「API 最佳實踐」或公司準則…等。這種做法對於初次處理一些 API 的小型團隊來說是合理的。但是，這些最初期的準則可能有待完善。

隨著負責 API 的團隊越來越多，各種風格、經驗與觀點也隨之增加。維持團隊之間的一致性越來越難，不僅僅因為有些團隊不遵守公司發表的準則，也可能因為新團隊採用一組不同的現成產品，限制他們遵守最初準則的能力。或許是因為他們不在事件串流（event-streaming）環境中工作，因為他們以 XML 來支援呼叫 / 回應風格的 API。他們當然需要準則，但準則必須與他們的領域配合，並且符合他們的客戶的需求。

在 API 管理專案的這個「中間階段」裡，領導方式和準則必須從「關於如何設計和實作 API 的具體指引」，轉變成「關於 API 的生命週期和它們之間如何互動的一般性指引」。第 6 章和第 7 章有我們看過的，成功組織為 API 專案的中間階段所做的各種事情。

公司一定有任何團隊都要遵守的準則，但那些準則必須適合他們的問題領域，以及他們的 API 客戶的需求。隨著你的社群越來越大，你的多樣性也會增加，此時非常忌諱試圖消除這種多樣性。此時，你要將控制槓桿從發號施令（例如「所有的 API 都必須使用以下的 URL 格式…」），切換到提供指引（例如「在 HTTP 上運作的 API 都應該使用以下的 URL 模板之一…」）。

在這個 API 管理的「後期階段」，治理的視角甚至可以進一步放大，以關注 API 如何隨著時間的推移而彼此互動，以及你的 API 如何與你的市集和產業中的其他公司的 API 互動。第 9、10、11 章都反映了在未來維護健康且穩定的 API 生態系統所需的「大格局」思維。

如你所見，隨著專案範圍的擴展，你的準則也要相應地擴展。這對全球企業來說特別重要，因為各地的文化、語言和歷史都會深深地影響各地區的團隊思考、創造與解決問題的方式。

這帶來下一個關鍵元素：規模。

## 規模

建立和維護健康的 API 管理計畫的另一個大挑戰是，如何應對規模隨著時間的變化。上一節談過，增加團隊和這些團隊製作的 API 可能是一項挑戰。在執行期監測與管理 API 所需的程序也會隨著系統的成熟而改變。追蹤位於同一個地點的同一個團隊所建構的少數幾個 API 所需的工具，與追蹤遍布多個時區與國家的成千上百個 API 入口所需的工具，有很大差異。

本書將這種 API 管理層面稱為「園林（landscape）」。隨著專案的擴展，你必須設法關注很多地方的很多團隊的很多流程，你更需要在執行期監測行為，以隨時了解系統的健康程度。本書的第二個部分（從第 9 章開始）將探討 API 園林管理概念如何協助釐清哪些元素值得關注，以及哪些工具與程序可以協助你掌握不斷成長的 API 平台。

API 園林帶來一系列的新挑戰。當你需要擴展生態系統時，用來設計、實作、維護單一 API 的程序不一定相同。基本上，這是一場數字遊戲：系統的 API 越多，它們互動的機會越高，有些互動導致意外行為（或「錯誤」）的可能性也就越高。大型系統就是這樣運作的，它有更多互動，以及更多意外結果。試圖消除這些意外不是治本之道，因為你不可能消滅所有的 bug。

這就帶來多數成長中的 API 專案所遇到的第三項挑戰：如何在 API 專案中執行適當的標準來減少意外的變動？

## 標準

當你開始進行園林規模的管理，而不是在 API 層面上時，標準（standard）引導團隊一致性地設計、實作與部署 API 的效果有關鍵的差異。

隨著團隊規模的擴大（包括負責你的組織的 API 的團隊），你的協調成本也會增加（見第 16 頁的「決策」）。擴大規模需要改變範圍。處理這項挑戰的重點是更依賴一般性標準，而不是具體的設計約束。

例如，全球資訊網從 1990 年問世以來，能夠持續良好運行的原因之一在於，它的設計者早就決定採取通用標準，該標準適用於任何軟體平台與語言，它不是專為單一語言或框架建立嚴格的實作指引。這使得創新團隊能夠發明新的語言、架構模式，甚至執行期框架，而不會破壞既有的實作。

這個協助 web 持續成功的長期標準有一個共同主軸，就是將元件與系統之間的互動標準化。web 標準的目的是讓各界能夠在網路上輕鬆地互相了解，而不是硬性規定如何實作組件的內在（例如，使用這個程式庫、這個資料模型…等）。同樣的，隨著你的 API 專案發展到一定的成熟度，你為 API 社群提供的指引應更加著重一般性的互動標準，而不是具體的實作細節。

這個轉變可能是艱辛的過程，但它對打造健康的 API 園林而言至關重要，在這種環境下，團隊可以建構能和現在與未來的 API 輕鬆互動的 API。

## 管理 API 園林

本章開頭說過，API 管理領域有兩項重大的挑戰：管理單一 API 的生命週期，以及管理所有 API 組成的園林。在我們拜訪許多公司並研究一般性的 API 管理方法時，我們發現許多「管理單一 API」的故事版本。坊間有許多「生命週期」和「成熟度模型」，試圖幫助你識別和減輕「設計、建構和部署 API」時的挑戰。但我們發現關於 API 生態系統（我們稱之為園林）的指引並不多。

園林有它自己的挑戰、它自己的行為與傾向。設計單一 API 時考慮的事情，與支援數十、上百或上千個 API 時考慮的事情不一樣。在生態系統中，你會遇到大規模的新挑戰，這些挑戰不會發生在單一 API 實例或實作中。本書稍後將更深入探討 API 園林，但是我們想在本書的開頭指出 API 園林給 API 管理帶來的三種獨特挑戰：

- 技術的擴展
- 團隊的擴展
- 治理的擴展

讓我們花點時間來回顧一下 API 管理中，與園林有關的層面。

### 技術

當你初次開始進行 API 專案時，有一系列的技術決策將會影響你的所有 API。你的「所有」 API 只有兩三個並不是重點，重點是，當你建立最初的 API 專案時，你必須有一套一致的工具和技術可以依靠。你會在我們討論 API 生命週期（第 7 章）與 API 成熟度時看到，API 專案並不便宜，你必須仔細地監測你投入多少時間和精力在這種活動上：對 API 的成功有很大的影響，但不需要過早投入大量資本的活動。這通常意味著選擇與支援

一小套工具，並提供明確的、詳細的指引文件，以協助團隊設計與建構既能解決商業問題，又能互相合作的 API。換句話說，你可以藉由限制技術的範圍，在早期取得成功。

因為我們提到的所有原因，這種做法在一開始很管用。但是隨著專案數量（見第 218 頁的「數量」）、和範圍的擴大（例如有更多團隊建構更多 API，在更多地點為更多業務提供服務…等），挑戰也會改變。當你發展 API 專案時，依靠一套有限的工具和技術可能變成減緩速度的關鍵。雖然在團隊規模不大時限制選擇可以加快進度，但是對大規模的團隊施加限制是既昂貴且危險的做法，當你在遙遠的地點增加團隊，並（或）接受新的業務部門或收購新公司來增加你的 API 園林時，情況更是如此。此時，多樣性（見第 213 頁的「多樣性」）對生態系統而言是更重要的成功驅動因素。

所以，管理 API 園林技術的重點在於，判斷園林何時已經大到可以開始增加技術的種類，而不是限制它們。其中有些技術與既有實作的現況有關，如果你的 API 園林需要支援你的組織既有的 TCP/IP SOAP 服務，你不能要求這些服務都使用你為新的 HTTP CRUD API 專案建立的 URL 指引。為新的事件驅動 Angular 實作、或舊的遠端程序呼叫（RPC）實作建立服務時也是如此。

範圍越廣，代表園林的技術種類越多。

## 團隊

專案的成長為 API 管理帶來的新挑戰不是只有技術面而已。團隊本身的組成也需要隨著園林的變動而調整。在 API 專案剛啟動的時候，你可以和少數幾位下定決心的伙伴一起做幾乎所有事情。此時，你會聽到「full-stack developer」或「MEAN developer」或其他類似的稱呼，意思是同一位開發者擁有 API 專案的所有層面的技能（MEAN 是 MongoDB、Express.js、Angular.js、Node.js 的縮寫）。你可能也聽過「初創團隊」或「獨立團隊」，它們都代表一個擁有所有技能的團隊。

當你的 API 很少，而且都用同一組工具來設計與實作時，你可以採取這種做法（見第 11 頁的「技術」），但是隨著 API 的規模與範圍變大，建構與維護 API 所需的技術種類也會增加，你不能指望每個 API 團隊都由一群擅長設計、資料庫、後端、前端、測試與部署的成員組成，你可能會指派一個團隊負責設計與建立一個以資料為中心的儀表板介面，來讓其他團隊使用。他們可能必須駕馭公司的所有資料格式，以及收集資料的工具。或者，你可能會讓一個團隊負責用單一技術來建構行動 APP（例如使用 GraphQL 或某些其他的查詢程式庫）。隨著技術種類的增加，你的團隊可能必須變得更專業化。我們會在第 8 章詳細討論這個主題。

隨著 API 園林的成長，團隊也要改變他們參與日常決策過程的方式。當團隊還很小，而且成員的經驗還不深時，你可以將決策權集中到一個指導小組。在大型的組織中，這種團隊通常稱為 Enterprise Architecture 團隊或其他類似的名稱。這種做法在規模與範圍較小時有效，但是當生態系統的同質性變低且範圍變廣時，這種做法就會造成麻煩。隨著技術越來越多，單一團隊不太可能掌握每一個工具與框架的細節。而且隨著團隊數量的增加，你要將決策權下放，因為中央單位應該無法了解全球企業的日常營運狀況。

解決這種問題的做法是將決策程序拆成所謂的*決策元素*（見第 25 頁的「決策的元素」），並將這些元素分配給公司內部的適當階級。為了讓生態系統持續成長，各個團隊必須在技術層面上更專業化，並且在決策層面上肩負更多責任。

## 治理

關於 API 園林的挑戰，我們想談的最後一個領域是*治理* API 專案的一般做法。與之前的其他案例一樣，我們發現，隨著生態系統的成長，治理的角色與手段也會改變。你將遇到新的挑戰，而且舊方法也不像過往那麼有效。事實上，頑固地採用舊治理模式可能會減緩甚至阻礙 API 的成功，特別是在企業層面上。

如同任何領導領域，當規模與範圍有限時，最有效的做法是提供直接指導，對小型的團隊而言如此，對新團隊往往也是如此。如果團隊沒有太多操作經驗，透過詳細的指導與（或）流程文件來傳授經驗可快速成功。例如，我們發現早期的 API 專案治理往往使用多頁的流程文件來解釋具體的工作，例如，如何為 API 設計 URL，或 URL 可以使用哪些名稱，或版本號碼應該放在 HTTP 標頭的哪裡。提供明確的指引以及少量的選項使得開發者難以偏離你的 API 實作方式。

但同樣的，隨著專案的成長，以及加入更多團隊與支援更多商業領域，社群的規模與範圍將開始讓你難以維護適合所有團隊的單一指導文件。雖然你可以將編寫與維護流程文件的工作「外包」出去，但這通常不是好主意，正如我們在第 11 頁的「技術」中談到的，技術的多樣性在大型的生態系統中是一種優勢，在企業治理層面上控制它會減緩專案的進展。

這就是為什麼隨著 API 園林的擴大，你的治理文件也要改變口氣，從直接提供流程指示，變成提供一般原則。例如，與其在文件中說明何謂有效的 URL，更好的做法是引導開發者在 Internet Engineering Task Force 查閱關於 URI 設計與所有權的指南（RFC 7320），並提供執行這個公共標準的一般性指導。你也可以在大部分的 UI/UX 指南中找到這種*原則性指南*的案例，例如 Nielsen Norman Group 的「10 Usability Heuristics for User Interface Design」（*https://oreil.ly/qU66X*）。這類文件提供很多選擇，並且指出為何要使用一種 UI

模式，而不是另一種模式。它們讓開發者與設計師知道為何要使用某個東西，以及使用它們的時機，而不是直接規定必須遵守的規則。

最後，對於大型組織，尤其是在多個地點和時區營運的公司，治理需要從分發原則轉為收集建議。這實質上顛覆了典型的中央治理模式。中央治理委員會的主要職責不是告訴團隊該怎麼做，而是從現場收集經驗資訊，找出相關性，並在更廣泛的組織裡回饋「最佳做法」指引。

所以，隨著 API 園林的成長，你的 API 治理模式也要從「直接提供建議」轉換成「提出一般原則」，再轉換成「收集和分享公司內部有經驗的團隊的做法」。我們將在第 2 章看到，你可以用一些原則和實踐法來建立適合貴公司的治理模式。

## 結論

在開頭的這一章，我們談了本書即將介紹的 API 管理重要層面。我們承認，雖然 API 將繼續成為一種推動力，但是只有 50% 的受訪公司相信自己有能力正確地管理 API。我們也釐清了「API」一詞的多種用法，以及那些用法為何讓你難以提供一致的專案治理模式。

而且，最重要的是，我們提到，管理「一個 API」與管理「API 園林」非常不同。管理一個 API 時，你可以依靠 AaaP、API 生命週期與 API 成熟度模型。API 的變動管理也非常注重這種「單一 API」的思維方式。但是，這只是故事的一部分。

接著，我們討論了 API 園林的管理，園林就是你的組織內的整個 API 生態系統。管理持續成長的 API 園林需要各種技術與指標，用來處理多樣性、數量、波動性、脆弱性與其他層面。事實上，這些園林層面都會影響 API 生命週期，本書稍後會再討論它們。

最後，我們談到，就連做出 API 專案決策的方式，也需要隨著時間而改變。隨著系統的發展，你也要下放決策權，就像分配 IT 元素一樣（例如資料儲存體、計算能力、資安，以及公司的基礎設施的其他部分）。

有了這個介紹的背景之後，我們接下來要先討論治理的概念，以及如何將決策與決策權的分配，當成整個 API 管理法的主要元素。