



後端熱門開發語言與 **GO** 比較

Gordan_Kuo

Compiler (編譯式語言)

- 代表語言： C, C++, C#
- 程式會先全部經過編譯器編譯成機器碼再轉換成執行檔執行
- 優點：
 1. 執行速度快
- 缺點：
 1. 因須先進行編譯才能執行程式，開發與除錯時間相對較長
 2. 因編譯流程中，需與本地 CPU 架構做對應，故不適合跨平台開發

Interpreter (直譯式語言)

- 代表語言: Python, JavaScript(node.js), PHP
- 運用直譯器一句一句轉譯後執行，不會一次直譯整份程式
- 優點:
 1. 可以跨平台執行
 2. 可以邊執行邊除錯，加速開發速度與降低維護難度
- 缺點:
 1. 執行上較無效率

GO 簡介

- Compiler language
- 由 google 創立與背書
- 內建多執行緒與並發，可同時執行多種功能與函式（搭配 **Goroutines**）適合擁有繁重的分散式系統架構使用
- 屬於靜態編譯程式語言，更容易發現錯誤及提升程式安全性
- 語法架構較簡潔（相對於 C 與 C # 等同為 **complier** 語言而言）
- 提供全面的內建函式庫，可降低使用第三方套件，免除相容性與安全性等問題
- 提供多樣編譯便利性（如：提供 **go fmt** 功能，可格式化程式編排，提供更佳的可讀性與可維護性；提供 **go module** 功能，方便套件管理；提供 **go test** 功能可以多件同時測試等）

PHP 簡介

- Interpreter language
- 歷史悠久（ since 1995 ），所以框架、社群等資源豐富
- 主要用於網頁開發（全球有 78.9% 的網站使用 PHP 架設）且可以嵌入 html 內執行
- 語法簡單，容易學習（擷取 c, java, perl 特性發展而來）
- 可以用 C, C++ 進行拓展
- 擁有許多自動化工具 for 測試以及部署
- 以物件導向及函式程式設計
- 屬於動態編譯程式語言，可加速程式編寫速度

PHP 與 GO 比較

- PHP > GO

- PHP 因歷史悠久，故社群方面以及相關資料較豐富
- GO 相較於 PHP 額外的函式庫支援 SDK 等較不健全
- GO 官方社群意見回饋較不積極甚至可能被取消（即便有問題）
- 因 PHP 開源特性，自由度較高

- PHP < GO

- GO 資料處理速度更快
- GO 上線部署時間更快
- GO 安全性更高
- 微服務下，GO 執行更少 Docker，且 API 能處理更高負載

Python 簡介

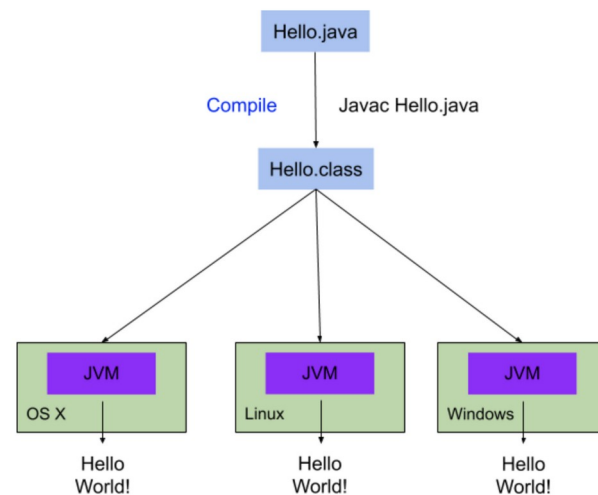
- Interpreter language
- 語法簡潔，易懂易學易讀
- 方便部署，高可擴充性
- 傑出的函式庫以及社群支持，提供多樣的開源軟體
- 物件導向設計
- 易於將資料 **scale** 化
- 快速建立模型，且方便測試
- 可在多種平台及系統提供協助與應用
- 屬於動態編譯程式語言，可加速程式編寫速度

Python 與 GO 比較

- Python>GO
 - Python 強大的函式庫及 **scale** 能力，更利於資料處理與分析、人工智慧、深度學習等
 - Python 文法簡單，易入門
- Python<GO
 - GO 效能及運算較佳
 - GO 支援併發，更適用大型專案
 - GO 偵錯能力更佳

Java 簡介

- Compiler and Interpreter language (JVM)
- 眾多開源軟體與函式庫可使用
- 物件導向設計
- 獨立平台運行 (JVM) 可再多種平台及系統使用
- 自動記憶體定位及垃圾回收機制
- 適合分散式運算並支援多線程
- 取消指向功能，不直接使用記憶體資料及引入安全管理機制，定義通道的類別，以提升安全性
- 提供多樣 API，如資料庫連接、網路連接、XML 轉換
- Android 應用程式開發語言



JAVA 與 GO 比較

- JAVA>GO
 - GO 須為每個平台建立二進位檔案，Java 則否，故Java 耗時更短
 - GO 無多型，在同一 `package` 裡面兩函式使用不同引數，但含義相同，仍須另外建立新方法名稱
 - GO 無原生繼承，需用 `interface` 等組合方式實現繼承
- JAVA<GO
 - Garbage collection 耗費時間更短，設定更簡易
 - GO 依賴度較低，佔據記憶體小，沒有複雜的注意事項 (JVM)
 - GO CPU 使用效率較佳
 - GO 語法簡潔，學習與開發的速度較快

C/C++ 簡介

- Compiler language
- 底層語言
- 可簡單設置阻擋及隱藏物件等功能
- 可快速執行且穩定
- 輕便，佔據記憶體小
- 可與系統硬體緊密運行
- 豐富函式庫與資源 (C++)
- 多樣應用範圍，如遊戲，GUI 應用，數學模型建立

C/C++ 與 GO 比較

- C/C++ > GO
 - C++ 支持繼承
 - 社群與資源 C/C++ 較豐富
 - C/C++ 性能速度較快
- C/C++ < GO
 - GO 語法簡潔易懂，編譯速度快
 - 相對於 C/C++ 需引用其他函式庫支持併發，GO 原生支持且易用
 - C 原生函式庫弱，C++ 大而不全，GO 較完善
 - C/C++ 不支持 Garbage collection，GO 有
 - GO 更適合網路環境與雲端應用，開發速度更快

R 簡介

- Interpreter language
- 全面的靜態分析語言
- 適合機器學習與資料分析
- 可以於各 OS 系統上無縫執行
- 高可擴展性
- 輕巧
- 提供多種 **package** 供使用，尤其是數據分析統計、機器學習、繪圖等
- 提供專屬免費 **RStudio IDE** 可使用

R 與 GO 比較


- R>GO
 - R 提供大量圖表與分析功能
 - R 的 package 完善（尤其與統計相關）
- R<GO
 - R 語法較不嚴謹，且相異於一般程式語言（如索引以”1”開始，賦值符號為”<-”），較難維護
 - R 開發效率不佳
 - R 安全性不佳
 - R 記憶體控制不佳

JavaScript 簡介

- Interpreter language
- 前端與後端語言 (搭配 node.js)
- 前後端皆使用同一種語言，可提昇團隊溝通效率
- 可跨平台開發、部署、運行
- 輕巧
- 完善網頁開發相關函式庫與模組
- 社群資源完善
- 網路服務效能佳
- 入門簡單

JavaScript(node.js) 與 GO 比較

- JavaScript > GO
 - JS 社群資源豐富，方便查詢引用
 - JS 第三方函式庫眾多
- JavaScript < GO
 - JS 不支援多執行緒，於大規模擴充套件並需處理多個並行任務的情境下容易出現錯誤
 - 非網路服務或數據庫傳輸等領域 JS 效能較差



感謝撥空查閱