00 开篇词 Go 为开发者的需求设计，带你实现高效工作

你好，我是**飞雪无情**，在技术领域从业近 10 年，目前在一家互联网公司担任技术总监，负责技术管理和架构设计。

2014 年，我因为 Docker 接触了 Go 语言，其简洁的语法、高效的开发效率和语言层面上的并发支持深深地吸引了我。经过不断地学习和实践，我对 Go 语言有了更深入的了解，不久后，便带领团队转型 Go 语言开发，提升了团队开发效率和系统性能，降低了用人成本。

在带领团队转型 Go 语言的过程中，我不断把自己学习 Go 语言的经验沉淀成文章，方便大家利用碎片时间学习，于是“飞雪无情”的公众号和知乎号就诞生了。现在，我已经发布了 200 多篇相关内容，在帮助数万名朋友有效学习 Go 的同时，还有幸拿到了知乎 Go 语言专题的最高赞。

Go 语言为开发者的需求而设计

K8s、Docker、etcd 这类耳熟能详的工具，就是用 Go 语言开发的，而且很多大公司（如腾讯、字节跳动等）都在把原来 C/C++、Python、PHP 的技术栈迁往 Go 语言。

在我看来，Go 作为一门高效率的工业化语言备受推崇，这与其语言本身的优势有直接的关系：

- 语法简洁，相比其他语言更容易上手，开发效率更高；
- 自带垃圾回收（GC），不用再手动申请释放内存，能够有效避免 Bug，提高性能；
- 语言层面的并发支持，让你很容易开发出高性能的程序；
- 提供的标准库强大，第三方库也足够丰富，可以拿来即用，提高开发效率；
- 可通过静态编译直接生成一个可执行文件，运行时不依赖其他库，部署方便，可伸缩能力强；
- 提供跨平台支持，很容易编译出跨各个系统平台直接运行的程序。

对比其他语言，Go 的优势也显著。比如 Java 虽然具备垃圾回收功能，但它是解释型语言，需要安装 JVM 虚拟机才能运行；C 语言虽然不用解释，可以直接编译运行，但是它不

具备垃圾回收功能，需要开发者自己管理内存的申请和释放，容易出问题。而 Go 语言具备了两者的优势。

如今微服务和云原生已经成为一种趋势，而 Go 作为一款高性能的编译型语言，最适合承载落地微服务的实现 ，又容易生成跨平台的可执行文件，相比其他编程语言更容易部署在 Docker 容器中，实现灵活的自动伸缩服务。

总体来看，**Go 语言的整体设计理念就是以软件工程为目的的，也就是说它不是为了编程语言本身多么强大而设计，而是为了开发者更好地研发、管理软件工程，一切都是为了开发者着想。**

如果你是有 1~3 年经验的其他语言开发者（如 Python、PHP、C/C++），Go 的学习会比较容易，因为编程语言的很多概念相通。而如果你是有基本计算机知识但无开发经验的小白，Go 也适合尽早学习，吃透它有助于加深你对编程语言的理解，也更有职业竞争力。

而在我与 Go 语言学习者进行交流，以及面试的过程中，也**发现了一些典型问题，可概括为如下三点**：

第一，学习者所学知识过于零碎，缺乏系统性，并且不是太深入，导致写不出高效的程序，也难以在面试中胜出。比如，我**面试**时常问字符串拼接的效率问题，这个问题会牵涉到 + 加号运算符、buffer 拼接、build 拼接、并发安全等知识点，但应聘者通常只能答出最浅显的内容，缺乏对语言逻辑的深层思考。

第二，很多入门者已有其他语言基础，很难转换语言思维模式，而且 Go 的设计者还做了很多相比其他语言的改进和创新。作为从 Java 转到 Go 语言的过来人，我非常理解这种情况，比如对于错误的处理，Java 语言使用 Exception，而 Go 语言则通过函数返回 error，这会让人很不习惯。

第三，没有开源的、适合练手的项目。

在过去分享 Go 语言知识的过程中，我融入了应对上述问题的方法并得到好评，比如有用户称“你的文章给我拨云见日的感觉！”“通过你的文章终于懂 context 的用法了！”……这些正向评价更坚定了我分享内容的信心。

于是在经过不断地思考、整理后，我希望设计更有系统性、也更通俗易懂的一门专栏。我的目标是通过这门课程帮助你少走弯路，比其他人更快一步提升职场竞争力。

这门课的亮点和设计思路

- **系统性设计**：从基础知识、底层原理到实战，让你不仅可以学会使用，还能从语言自身的逻辑、框架层面分析问题，并做到能上手项目。这样当出现问题时，你可以不再盲目地搜索知识点。
- **案例实操**：我设计了很多便于运用知识点的代码示例，还特意站在学习者的视角，演示了一些容易出 Bug 的场景，帮你避雷。我还引入了很多生活化的场景，比如用枪响后

才能赛跑的例子演示 sync.Cond 的使用，帮你加深印象，缓解语言学习的枯燥感。

- **贴近实际**：我所策划的内容来源于众多学习者的反馈，在不断地交流中，我总结了他们问题的共性和不同，并有针对性地融入专栏。

那我是怎么划分这门课的呢？

作为初学者，不管你是否有编程经验，都需要先学习 Go 语言的基本语法，然后我会在此基础上再向你介绍 Go 语言的核心特性——并发，这也是 Go 最自豪的功能。其基于协程的并发，比我们平时使用的线程并发更轻量，可以随意地在一台普通的电脑上启动成百上千个协程，成本非常低。

掌握了基本知识后，我们来通过底层分析深入理解原理。我会结合源码，并对比其他语言的同类知识，带你理解 Go 的设计思路和底层语言逻辑。

此时你可能还有一些疑惑，比如不知道如何把知识与实际工作结合起来，所以就需要 Go 语言工程质量管理方面的知识了。而最后，我会用两个实战帮你快速上手项目，巩固知识。

所以，我根据这个思路将这门课划分成 5 个模块：

- **模块一：Go 语言快速入门**：我挑选了变量、常量等数据类型、函数和方法、结构体和接口等知识点介绍，这部分内容相对简洁，但已经足够你掌握 Go 的基本程序结构。
- **模块二：Go 语言高效并发**：主要介绍 goroutine、channel、同步原语等知识，让你对 Go 语言层面的并发支持有更深入的理解，并且可以编写自己的 Go 并发程序设计。最后还会有一节课专门介绍常用的并发模式，可以拿来即用，更好地控制并发。
- **模块三：Go 语言深入理解**：Go 语言底层原理的讲解和高级功能的介绍，比如 slice 的底层是怎样的，为什么这么高效等。这个模块也是我特意设计的，我在初学编程时，也有只学习如何使用，而不想研究底层原理的情况，导致工作遇到障碍后又不得不回头恶补，后来发现这是初学者的通病。但理解了底层原理后，你才能灵活编写程序、高效应对问题。
- **模块四：Go 语言工程管理**：学习一门语言，不光要掌握它本身的知识，还要会模块管理、性能优化等周边技能，因为这些技能可以帮助你更好地进行多人协作，提高开发效率，写出更高质量的代码。你可以在这个模块学到如何测试 Go 语言以提高代码质量、如何做好性能优化、如何使用第三方库提高自己项目的开发效率等。
- **模块五：Go 语言实战**：Go 语言更适合的场景就是网络服务和并发，通过开发 HTTP 服务和 RPC 服务这两个实战，可以把前四个模块的知识运用起来，快速上手。

作者寄语

我一直不厌其烦地跟团队小伙伴说，Go 语言是一门现代编程语言，相比其他编程语言，它对我们开发者有更好的用户体验，因为它的目的就是让我们更专注于自己业务的实现，提

高开发效率。与此同时，当下的云原生是一种趋势， Go 语言非常适合部署在这种环境中，越早学习越有竞争力。

此外，我在上文中也反复强调了学习底层原理的重要性。编程语言有很多共通之处（比如概念、关键字、特性语法等），吃透后再学习其他的编程语言会简单得多，原因在于你理解了语言本身。所以在学习 Go 语言的过程中，我希望你多想、多练，深入理解，融会贯通。

现在，跟我一起踏上 Go 语言学习之旅吧，Let's Go！

下一页