07｜目标：站在巨人肩膀，你的理想框架到底长什么样？

2021-09-27 叶剑峰

《手把手带你写一个Web框架》 课程介绍 >

讲述：叶剑峰
时长 15:43 大小 14.40M

你好，我是轩脉刃。

在前面几节课，我们使用 Golang 的 net/http 库实现了一个带有控制器、路由、中间件等功能的 Web 框架，凡事都在向完成一个自定义 Web 框架的方向发展。

在开发的过程中，不知道你有没有意识到，其实框架中的某个模块，比如说路由，实现的方法不止一种，每一个模块要实现的功能也各有不同，所以用哪一种方法来实现，以及要实现哪些功能，都是一种选择。

而每种选择的背后，其实都是方向问题，因为这些选择共同构成了一个框架的倾向性，也就是设计感。你要明白，我们的最终方向是：实现我们想要的理想框架。这就好比驾驶汽车的时候，作为司机，你要对目的地有明确清晰的认识。

那理想框架到底是什么样子的？这个终极问题，闭门造车是无法得到答案的，所以今天，我想让你先从埋头搭建 Web 框架的视角中暂时跳出来，站在更高的角度来纵观全局。

开源框架怎么比较

我们先进入开源世界，对比开源世界中现有的各种 Web 框架，理解一下它们的实现选择和意图。

Golang 语言的 Web 开发有很多的框架可以用，但是选择太多也不是好事，因为在技术群里我总会遇到群友有这些疑问：哪款框架比较好呢？我要选择哪款框架呢？这些疑问至少暴露出两个问题：一不知道如何比较开源框架、二不了解这些开源框架，那么接下来我们一一解答。

如何比较开源框架？说到比较，是一定要有标准的。但是衡量一个 Web 框架是多维度的事情，也没有定论。这里我按照优先级顺序，列出我衡量一个框架的评判标准，你可以参考。

标准	说明
核心模块	服务启动方式、路由分发机制是怎么样的 上下文context封装性如何、中间件机制是怎么设计
功能完备性	是否有提供日志模块、是否有提供命令行工具、是否有提供缓存机制等
框架扩展性	需要扩展某个功能，是否改动较大、是否支持功能实现的可插拔
框架性能	框架每秒能支持多少请求、是否有性能问题
文档完备度/社区活跃度	是否有完善的文档支持、社区是否足够活跃、咨询问题多久能得到回复

极客时间

核心模块

一个框架最核心的几个部分就是我们前几节课讲到的：HTTP 服务的启动方式、路由、Context、中间件、重启方式，它们的实现非常关键，往往影响到整个框架的表现，也能直接体现出框架作者的开发水平。

框架的核心模块就好像是汽车的引擎，一旦核心模块出了问题，或者有隐性的缺陷，后果往往是无法弥补的。理想的核心模块必须要有设计感，有自己的思想，代码质量、性能都不能出问题。

功能完备性

框架最好能尽可能多地提供功能或者规范，比如有自己的日志模块、脚手架、命令行工具，甚至自己的 ORM、缓存等等。

要知道，框架的本质还是在于提高开发效率，在团队中，我们**希望不同水平的同学能写出基本一样的代码，那就要靠框架这个顶层设计来规范了。**试想一下，如果框架中提供了很方便的参数验证规范，那在开发应用的时候，还有谁愿意走解析和正则匹配来验证参数呢?

在功能完备性的选择上，我们往往会根据之后是否希望自定义需求来确定。这里说的自定义需求，指的是定制自己的日志、ORM 等模块的需求。

框架扩展性

理想的框架，它的扩展性一定要好。**框架要做的事情应该是定义模块与模块之间的交互标准，而不应该直接定义模块的具体实现方式。**我可以在这个标准上扩展出我需要的功能，这样整个框架才会比较灵活。

Web 领域的技术边界在不断扩展，谁也无法保证框架中所用的库，哪怕是最基本的日志库，能永远满足需求。当框架使用者想为应用增加一个功能，或者替换某个第三方库的时候，如果改动的地方非常多，要大动干戈，甚至根本就无法支持替换和增加，那这个框架的扩展性就比较弱了。

框架性能

虽然大部分框架都是封装 net/http，但是封装程度不同以及具体的实现选择不同（使用的路由匹配、上下文机制等），就会有不同的性能表现。

我们选择框架之后，最终是要在框架上运行代码的，如果运行效率有指数级别的差距，是不能接受的。市面上所有框架的性能，你可以参考这个第三方测评结果。

其实对框架性能来说，大部分场景里，我们是不会有极致性能需求的。所以看各种框架的性能评估，**我们不应该把各个框架孤立出来看，应该将差不多量级的性能归为一组**。不要去比较单个框架的性能差异，而应该去比较不同量级组之间的性能差异，因为在相同量级下，其实框架和框架之间的性能差别并不是很大。

文档完备度及社区活跃度

开源不仅仅是将代码分享出去，同时分享的还有使用文档，官方必须提供足够的介绍资料，包括文档、视频、demo 等。

文档和社区是需要不断运营的，因为在使用的过程中，我们一定会遇到各种各样的问题，官方的回复以及一个活跃的社区是保障问题能得到解决的必要条件。

所以当我们选择一个框架的时候，官网、GitHub 的 star、issue 都是很好的衡量标准。当选择框架之后，**不妨每天花一点时间在这个项目的官网、GitHub 或者邮件组上，你会得到很多真实信息**。

现在基本了解这五点评判标准了，我们可以用这把尺子衡量一下市面上的框架，帮助你迅速熟悉起来。

比较开源框架

在开源社区有个 go-web-framework-stars 项目，把 41 款 Go Web 框架按流行程度做了个列表，其中 Gin、Beego、Echo 这三个框架是 Star 数排名前三的，下面我们就针对这三个框架来分析。

在刚才的五个维度中，我们会重点分析这三个框架的核心模块和功能完备性这两点，另外三点扩展性、性能以及文档完备性资料很多，我只在表格中简单说明一下。

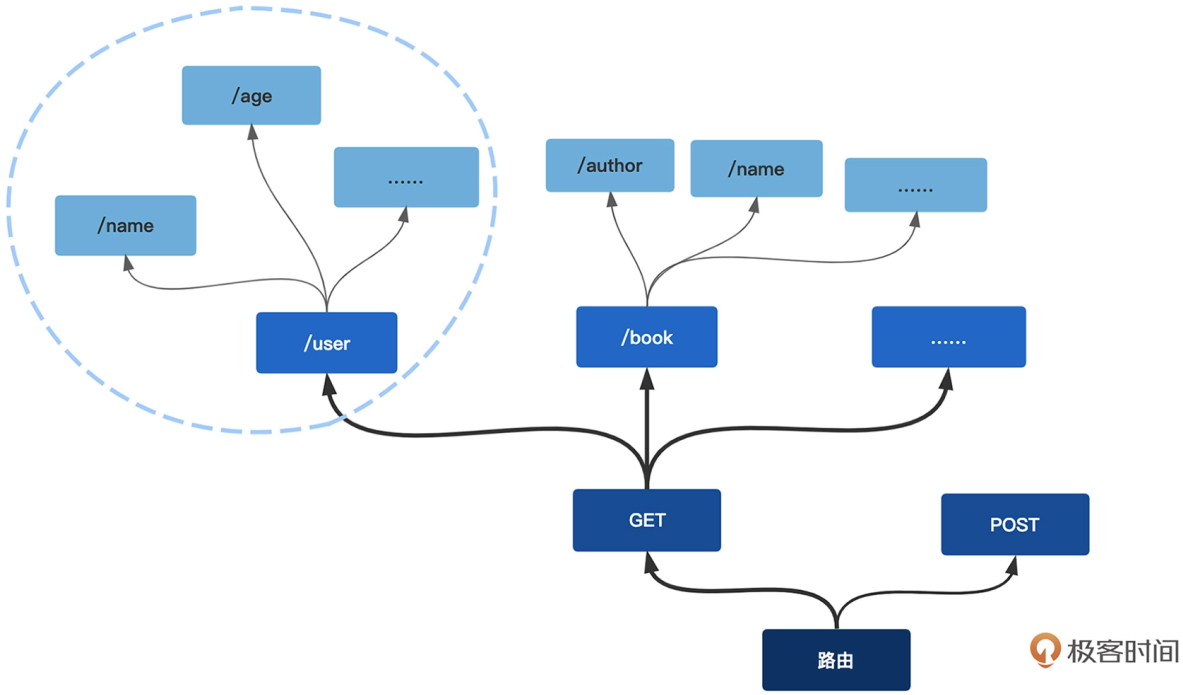
Beego

Beego 是一款国人开发的 Web 框架，它出现得非常早，中间经历过一次比较大的改版，目前版本定位在 2.0.0。

先来看 Beego 的核心模块，我们还是从 HTTP 服务的启动方式、路由、Context、中间件、重启方式这五个方面来分析。

Beego 提供多种服务启动方式：HTTP/HTTPS、CGI、Graceful。

它的路由原理是为每个 HTTP 方法建立一个路由树，这个路由树结构的每个叶子节点是具体的执行控制器，每个中间节点是 HTTP 请求中斜杠“/”分隔的节。比如 /user/name 从根到叶子节点的通路就是：1 个根节点、1 个中间节点 user、一个叶子节点 name。



Beego 的自动匹配路由和注解路由是比较有特色的，这里我简单解释一下：

> 自动匹配路由的意思是，如果我们把一个控制器注册到自动路由中，路由寻址的时候，会根据“控制器名称 + 控制器类”中的方法名，自动进行映射匹配。
>
> 注解路由是指在注册路由树的时候，会根据控制器类中某个方法的注释来解析并创建路由树。

Context 的设计方面，Beego 的 Context 是藏在 controller 结构中的，而不是每个函数的第一个参数带着 Context，这个和现在 Golang 提倡的“函数第一个参数为 Context”的规范是有些不同的，可能也是因为 Beego 出现的时间比较早。

在功能完备性方面，Beego 框架提供了很多周边的功能。比如 Beego 提供了一个原生的 ORM 框架、自定义的 Logger 库、参数验证 Validate 库，甚至还默认提供了管理 GC、Goroutine 等管理接口。

标准	Beego	评分
核心模块	基于标准库，采用字典树进行路由设计； 提供自定义context，但context设计与标准库风格有一定差异； 提供中间件逻辑；提供多种重启方式	4分
功能完备性	提供ORM、Logger、Validator、Admin等功能，功能提供较为完备	5分
框架扩展性	部分模块可以扩展，但由于集成较多，扩展和替换成本比较高	3分
框架性能	参考第三方测评结果	4分
文档完备度/社区活跃度	文档较为完善，社区活跃度高	5分

总体来说，使用 Beego 的最大感受就是“全”。这是一个很全的框架，开发 Web 应用所需要的所有组件基本都能在这里找到，**如果选择它做业务开发，功能完备性是最重要的因素**。所以在从零开始，希望快速开发的场景中，我们会尽量选择使用这个框架。

Echo

Echo 框架目前最新版本是 v4.0.0。它的底层也是基于 net/http，但并没有提供类似 Beego 通过信号的 Graceful 启动方式，而是暴露标准库的 ShutDown 方法进行请求的关闭接口，具体使用的方式是开关还是配置，都交由使用者决定。

在路由方面，Echo 的路由也是使用字典树的结构。和 Beego 的树不同，它并不是为每个 HTTP 方法建立一个树，而是只整体建立一个树，在叶子节点中，根据不同的 HTTP 方法存放了不同的控制处理函数，所以你可以在它的叶子节点中看到如下这么一个 methodHandler 结构：

```
methodHandler struct {
    connect  HandlerFunc
    delete   HandlerFunc
    get      HandlerFunc
    head     HandlerFunc
    options  HandlerFunc
    patch    HandlerFunc
    post     HandlerFunc
    propfind HandlerFunc
    put      HandlerFunc
    trace    HandlerFunc
    report   HandlerFunc
}
```

Echo 也封装了自己的 Context 接口，提供了一系列像 JSON、Validate、Bind 等很好用的对 request 和 response 的封装。

相较于 Beego 库，Echo 库非常轻量，除了基本的路由、Context 之外，大部分功能都以 Middleware 的形式提供出来。Echo 的 Middleware 是一个 echo.MiddlewareFunc 结构体的无参数函数。在框架的 middleware 文件夹中，提供了诸如 logger、jwt、csrf 等中间件。

标准	Echo	评分
核心模块	基于标准库，采用字典树进行路由设计；提供自定义context； 提供中间件逻辑；提供简要启动方式	5分
功能完备性	功能完备性较差，核心框架仅提供是核心模块的封装，社区提供周边封装	3分
框架扩展性	框架使用中间件来进行功能扩展，制定了扩展规范，扩展性较好	5分
框架性能	参考第三方测评结果	5分
文档完备度/社区活跃度	文档较为完善，社区活跃度中等	5分

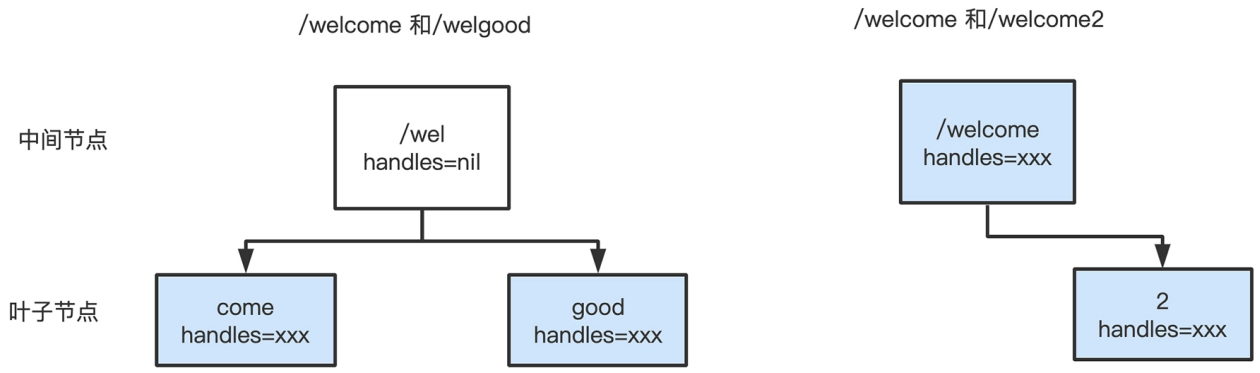
总体来说，**Echo 的使用者更看中扩展性和框架性能**。和定位一样，它是一种高性能、可扩展、轻量级的 Web 框架，但麻雀虽小，五脏俱全。目前看起来，它的社区活跃度比 Gin 框架稍差一些（从两者的中间件贡献数可以看出）。所以我感觉，Echo 框架更适合个人开发者，而且需要有一定的扩展框架能力。

Gin

Gin 框架目前最新版本 1.7.2。它和 Echo 一样属于非常轻量级的应用，基本实现的就是 Web 框架最核心的部分。

路由方面，和 Echo 一样使用字典树，但是又和 Echo/Beego 不一样的是，它的中间节点并不是斜杠分隔的请求路由，而是**相同的字符串部分**。

比如下图左边的例子，/welcome 和 /welgood 会创建一个中间节点 (/wel) 和两个叶子节点 (come) 和 (good)，而且并不是只有叶子节点才包含控制器处理函数，中间节点也可能存在控制器节点函数。比如下图的右边例子，/welcome 和 /welcome2 包含中间节点 (/welcome) 和叶子节点 (2)，中间节点 /welcome 也包含了一个处理函数。



Gin 的路由还有一个特色就是，**使用索引来加速子树查询效率**。再看图中左边的例子，/wel 中间节点还带着一个索引 cg，代表的是子树的第一个字母。所以路由查找的时候，会先去索引 cg 中查找，当前路由除“/wel”的第一个字母，是否在索引中，如果在，索引的下标序列就代表第几个子树节点。

不过，Gin 的路由并不是这个框架作者原创的，使用的是第三方的 httprouter 包。Gin 使用这个核心包的方法也不是直接 import 第三方包，而是将其 copy 进自己的代码库，当然可以这么用也是因为 httprouter 第三方库使用的是 BSD 这种有很大自由度的许可证协议。

Context 基本就和 Echo/Beego 的设计一样，自己封装了 Context 结构。但是，Gin 的 Context 结构实现了标准库定义的 Context 接口，即 Deadline、Done 等接口。所以按照 Golang 中定义的鸭子类型概念（长得像鸭子，那么就是鸭子），我们可以把 Gin 中的 Context 当作标准库的 Context 一样使用，这点在实际开发工作中是非常方便的。

在中间件上，Gin 框架没有定义所有的 middleware，而是定义了 middleware 函数：

复制代码

```
1 HandlerFunc func(*Context)
```

链式加载和调用 middleware，并将这个 middleware 的具体实现开放给社区，设计了社区贡献 organizations ，让社区的开源贡献者把自己实现的中间件统一放在这个 organizations 中，提供给所有人使用，而 Gin 的核心代码及相关则放在另一个 organizations 中。这种为开源社区制定标准，并且不断推进和审核开源贡献代码的做法，也是 Gin 社区活跃度如此之高的原因之一。

标准	Gin	评分
核心模块	基于标准库，采用字典树进行路由设计； 提供自定义context；提供中间件逻辑，提供简要启动方式	5分
功能完备性	功能完备性较差，核心框架仅提供是核心模块的封装，社区提供周边功能	4分
框架扩展性	框架使用中间件来进行功能扩展，制定了扩展规范，扩展性较好	5分
框架性能	参考第三方测评结果	5分
文档完备度/社区活跃度	文档较为完善，社区活跃度高	5分

极客时间

总体来说，**Gin 比较适合企业级团队使用**。它的社区活跃度较高，社区贡献的功能模块较多，能很好补充其功能完备性；同时 Gin 的扩展性也很好，我们可以在社区贡献模块和自研模块中做出很好的取舍。

这里我们只介绍了三个框架，之后你想要快速了解一个新框架，也可以参考这套思路。

你理想的框架是什么样的

现在你已经知道了如何比较开源框架，也对三款最受欢迎的开源框架有一定了解。那么我们再回答开头提到的终极问题：如果我们要做一款 Web 框架，它应该是什么样子?

其实从刚才的分析也可以看出来，**这个问题是见仁见智的，和你的工作经验、技术阅历、甚至技术的理念都有关系**。有的人追求的是世界上运行速度最快的框架；有的人追求的是灵活性最高的框架；有的人追求的是功能最全的框架。

具体你要根据自己的应用场景来选择，包括你 Web 应用的业务场景、并发需求、团队规模、工期等等。回看刚才提出的五个选择维度，以我们聊过的三个 Web 框架为例。

在保证框架的核心模块能满足要求的情况下，我们一般在功能完备性和框架扩展性之间取舍。

如果你开发一个运营管理后台，并发量基本在 100 以下，单机使用，开发的团队规模可能就 1～2 个人，那你**最应该考虑功能完备性，明显使用 Beego 的收益会远远大于使用 Gin 和 Echo**。因为如果你选择 Gin 和 Echo 之后，还会遇到比如应该选用哪种 ORM、应该选用哪种缓存等一系列问题，而这些在功能组件相当全面的 Beego 中早就被定义好了。

如果你有一定的团队规模，有比较高的并发量，而且你感觉后续对框架的改动需求或者扩展需求会很高，比如你希望自己开发一个适合团队使用的缓存方法。那么这个时候，**你应该把框架扩展性放在最高级，可能 Gin 和 Echo 更适合你**。如果你要更多的灵活性，你可能会考虑直接从 net/http 标准库开始，不使用任何的开源 Web 框架。

所以，选择框架信奉一个原则：只选最适合的，不选最贵的。如果你在几个框架中犹豫不定，除了可以用五个维度比较框架之外，你更应该多花时间内省思考清楚你的真正需求。

小结

我们今天尝试通过回答两个分问题，来思考一个终极问题，理想框架究竟应该是什么样的。

标准	说明
核心模块	服务启动方式、路由分发机制是怎么样的 上下文context封装性如何、中间件机制是怎么设计
功能完备性	是否有提供日志模块、是否有提供命令行工具、是否有提供缓存机制等
框架扩展性	需要扩展某个功能，是否改动较大、是否支持功能实现的可插拔
框架性能	框架每秒能支持多少请求、是否有性能问题
文档完备度/社区活跃度	是否有完善的文档支持、社区是否足够活跃、咨询问题多久能得到回复

对第一个分问题如何比较开源框架，我们提出了五个维度，按照优先级顺序依次为：核心模块、功能完备性、框架扩展性、框架性能、文档完备度及社区活跃度。然后从这五个维度，简要分析了现在最流行的三个开源框架 Beego、Gin 和 Echo。

最后我们回到终极问题，探讨我们理想中的框架应该是什么样子的？总结一句话就是，**在搞清楚真正的业务需求后，选最合适的框架就可以了**。把握好这一点，你今后在遇到框架选择问题的时候，就不会太迷茫。

思考题

我们比较了三个框架，但是你可能也用过其他的框架，不妨介绍一下你在过去工作中使用过的框架，以及使用感受？

欢迎在留言区分享你的思考。感谢你的收听，如果你觉得有收获，也欢迎你把今天的内容分享给你身边的朋友，邀他一起学习。我们下节课见～

上一篇 06｜重启：如何进行优雅关闭？

下一篇 08｜自研or借力，集成Gin替换已有核心（上）

精选留言

写留言

> 由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。