GoWeb编程

# 接收请求

在第二章中，我们看到了一个简单的网上论坛web应用是由什么组件构成的，也了解到了这些组件是如何组织成一个Go web应用的。虽然我们已经兑构成goweb应用的各个组件有了基本的了解，但关于这些组件还有很多值得深入的事情。在接下来的几章里，我们将更为深入地了解这些组件的细节，并详细地探讨这些组件是如何组合起来的。

本章和下一章，将兑Web应用的大脑（也就是负责接收和处理客户端请求的处理器）进行讨论。在本章中，我们将要学习的是如何使用Go语言去创建一个web服务器，以及如何处理客户端发送的请求。

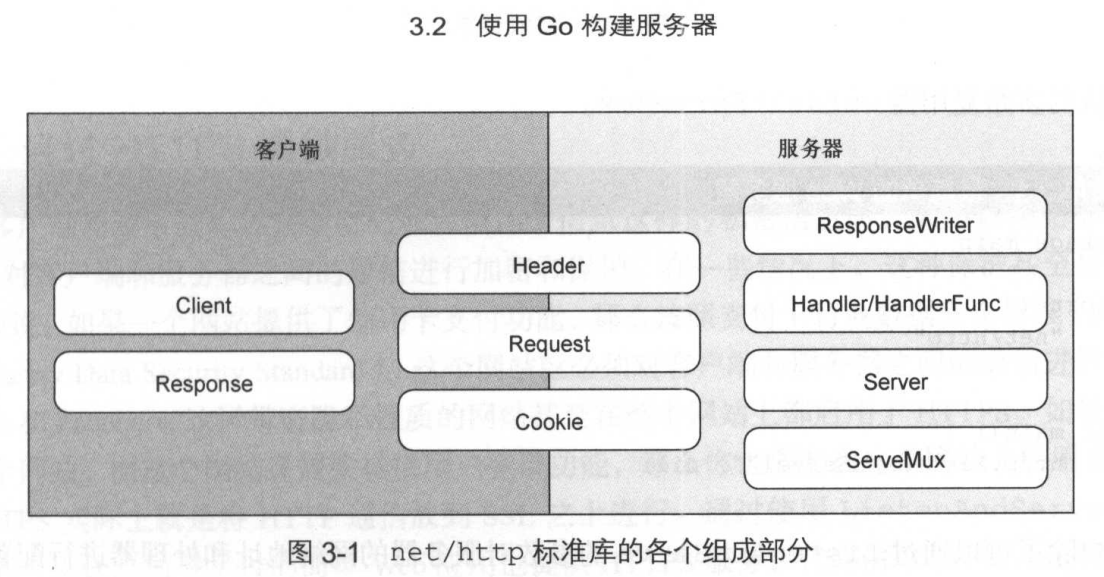
## Go的net/http标准库

在进行Web应用开发的时候，使用成熟的并且复杂的Web应用框架通常会使开发变得更加迅速和简便，但这也意味者开发者必须接收框架自身的一套约定和模式。虽然很多框架都认为自己提供的约定和模式是最佳实践，但是如果开发者没有正确的理解这些最佳实践，那么对最佳实践的应用就可能会发展为货物崇拜变成：开发者如果不了解这些约定和模式的用法，就可能会在不必要甚至有害的情况下盲目的使用它们。

举个例子来说，因为HTTP是一种无连接协议，通过这种协议发送给服务的请求对服务器之前处理过得请求一无所知，所以应用程序才会以cookie的方式在客户端实现数据持久化，并以会话的方式在服务器上实现数据持久化，而不了解这一点的人是很难理解为什么要在不同连接之间使用cookie和会话实现信息持久化的。为了降低使用cookie和会话带来的复杂性，web应用框架通常都会提供一个统一的接口，用于在连接之间实现持久化。这样做的结果是，很多新手程序员都会想当然地假设在连接之间进行持久化唯一要做的就是使用框架提供的接口。但是由于这类接口通常都是根据框架自身的习惯制定的，因此不同框架提供的接口可能会有所不同。更糟糕的是，不同的框架可能会提供一下名字相同的接口，但是这些同名接口之前的实现却又千差万别、各不相同，因此给开发带来不必要的困惑。通过这个例子可以看出，使用框架进行web应用开发意味着将框架与应用进行绑定，之后无论

对Go语言来说，隐藏在框架之下的通常是net/http和html/template这两个标准库，本章和接下来的第4涨将介绍net/http库，而之后的第5章将介绍html/template库。

如图3-1所示，net/http标准库可以分为客户端和服务器两个部分，库中的结构和函数有些只支持客户端和服务器这两者中的一个，而有些在同时支持客户端和服务器：



# 处理请求

## 请求和响应

### Request结构

Request结构表示一个客户端发生的Http请求报文，虽然HTTP请求包是有一系列文本行组成的，但request结构并部署完全按照报文逐字逐行定义的。实际情况是，这个结构只包含了报文在经过语法分析之后，其中较为重要的信息；除此之外，这个结构还有一系列相应的方法可供使用。