简书 发现 下载App 搜索 注册 登录 开通VIP

一日一学_Go语言HTTP Server源码分析

Shufan 作者 2019.11.08 22:01:51 字数 2 阅读 27

Go语言HTTP Server:

HTTP server，顾名思义，天然net/http的功能模块，HTTP等一个很有价值的模块，net/http的实现，使用过这样子的产品，我以用户端的浏览器和服务器的交互为例，

HTTP服务器单实现

```
package main
import (
"net/http"
"fmt"
)
func hello(w http.ResponseWriter, r *http.Request) {
fmt.Fprintf(w, "Hello, %q", r.URL.Path[1:])
}
func main() {
http.HandleFunc("/", hello)
http.ListenAndServe(":2000", nil)
}
```

你已经成功创建了HTTP server。

源码分析

以最简单的
http.HandleFunc("/", hello) 这句为例来分析源码

```
func HandleFunc(pattern string, handler func(ResponseWriter, *Request)) {
DefaultServeMux.HandleFunc(pattern, handler)
}
```

DefaultServeMux是什么？
DefaultServeMux是ServeMux的一个实例，
ServeMux又是什么？

```
// ServeMux is an HTTP request multiplexer.
// It matches the URL of each incoming request against a list of registered
// patterns and calls the handler for the pattern that
// most closely matches the URL.
//
// Patterns may optionally begin with a host name, restricting
// matches to URLs on that host.
//
// Patterns named fixed paths, rooted paths, or subtrees
// (ending in '/') also accept requests for paths
// that are not in the map, by redirecting to the
// corresponding path.
//
// Patterns may also specify a method, restricting
// matches to requests with that method.
//
// Patterns may also specify a host and path, restricting
// matches to requests for that host and path.
//
// Patterns ending in "/" match all paths not
// matched by other patterns.
//
// Patterns without a slash match only
// the exact path.
//
// Longer patterns take precedence over shorter ones,
// so that two overlapping patterns can coexist.
//
// Large web servers use many patterns and
// liberal use of subtree patterns to
// keep their lists of HandleFunc calls short.
//
// Matching against the URL path is done after
// cleaning the path to remove "." and ".."
// elements and to lowercase any upper-case
// letters in the host name.
//
// ServeMux also takes care of sanitizing the
// URL request path and the Request.Header
// Host, to stop common exploits.
//
// ServeMux also redirects requests for
// "/foo/bar/..." to "/foo/" if "/foo/" is
// a registered path.
//
// For high-profile or untrusted servers, a
// ServeMux may be too permissive. For
// instance, a single pattern ending in "/"
// will match every path in the server
// unless other patterns exist.
//
// ServeMux also takes care of rejecting
// requests with unparsable paths.
//
// The zero ServeMux is valid and
// ready to accept registrations.
//
// ServeMux is safe for concurrent use
// by multiple goroutines.
//
// ServeMux is not yet fully compatible
// with HTTP/1.1 specification.
type ServeMux struct {
mu    sync.RWmutex
m     map[string]muxEntry
hosts bool // whether any patterns contain hostnames
}
```

ServeMux定义最重点的两个成员变量mu，和m，其中mu了解为RWmutex锁和sync包，(sync的使用及原理请参考Mr.Chen的文章)，，，最后发现ServeMux的HTTP实现，并且暴露出（这一部分是Go语言的HTTP server的核心）
最后发现最重要的是HTTP Handler接口的定义？

```
// Handler interface
type Handler interface {
ServeHTTP(ResponseWriter, *Request)
}
```

这个接口，是HTTP server的一个核心部分

```
// A Request represents an HTTP request received by a server
// or to be sent by a client.
//
// The field semantics differ slightly between client and
// server usage. In addition to the notes on the fields below
// see the HTTP specification.
//
// Except for Body, no field may be updated while
// a request is being handled.
//
// Except for Body, Header, and URL, all
// properties are intended to be read-only
// during a request.
type Request struct {
// Method specifies the HTTP method (GET, POST, PUT, etc.).
// For client requests an empty string means GET.
//
// Go's HTTP client does not support sending a request
// with the CONNECT method.
//
// Go's HTTP server does not generally do any
// things with the CONNECT request
// other than send back an error.
//
// The value must be a valid HTTP token.
// It is not the method; it is a
// more general extension to the
// HTTP protocol.
//
// Go's HTTP client does not support
// the CONNECT method.
//
// Go's HTTP server does not generally
// do any things with the CONNECT
// request other than send back an
// error.
//
// The value must be a valid HTTP
// token. It is not the method;
// it is a more general extension
// to the
```