HTTP 编程

Web工作流程

Web服务器的工作原理可以简单地归纳为：

- 客户机通过TCP/IP协议建立到服务器的TCP连接
- 客户端向服务器发送HTTP协议请求包，请求服务器里的资源文档
- 服务器向客户机发送HTTP协议应答包，如果请求的资源包含有动态语言的内容，那么服务器会调用动态语言的解释引擎负责处理“动态内容”，并将处理得到的数据返回给客户端
- 客户机与服务器断开。由客户端解释HTML文档，在客户端屏幕上渲染图形结果

HTTP协议

超文本传输协议(HTTP，HyperText Transfer Protocol)是互联网上应用最为广泛的一种网络协议，它详细规定了浏览器和万维网服务器之间互相通信的规则，通过因特网传送万维网文档的数据传送协议。

HTTP协议通常承载于TCP协议之上。

Go语言通过引入net/http包来实现http网络访问，并提供HTTP客户端和服务端的实现。

HTTP客户端

基本的HTTP/HTTPS请求 Get、Head、Post和PostForm函数发出HTTP/HTTPS请求。

基本方法：

```
//发送get请求
resp, err := http.Get("http://baidu.com/")
...
//发送post请求
resp, err := http.Post("http://baidu.com/upload", "image/jpeg", &buf)
...
//发送postfrom请求
resp, err := http.PostForm("http://baozi.com/form",
    url.Values{"key": {"Value"}, "id": {"123"}})
```

程序在使用完response后**必须关闭回复的主体。**

```
//程序在使用完response后必须关闭回复的主体。
resp, err := http.Get("http://baidu.com/")
if err != nil {
    // handle error
}
defer resp.Body.Close()   //关闭body
body, err := ioutil.ReadAll(resp.Body)
// ...
```

示例：

```
// HTTP/client/client.go
```

```
// HTTP client端

package main

import (
    "fmt"
    "io"
    "net/http"
)

func main() {
    //resp, _ := http.Get("http://www.baidu.com")
    //fmt.Println(resp)
    resp, _ := http.Get("http://127.0.0.1:8000/go")
    defer resp.Body.Close()
    // 200 OK
    fmt.Println(resp.Status)
    fmt.Println(resp.Header)

    buf := make([]byte, 1024)
    for {
        // 接收服务端信息
        n, err := resp.Body.Read(buf)
        if err != nil && err != io.EOF {
            fmt.Println(err)
            return
        } else {
            fmt.Println("读取完毕")
            res := string(buf[:n])
            fmt.Println(res)
            break
        }
    }
}
```

HTTP服务端

ListenAndServe使用指定的监听地址和处理器启动一个HTTP服务端。处理器参数通常是nil，这表示采用包变量DefaultServeMux作为处理器。

Handle和HandleFunc函数可以向DefaultServeMux添加处理器。

```
http.Handle("/foo", fooHandler)
http.HandleFunc("/bar", func(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintf(w, "Hello, %q", html.EscapeString(r.URL.Path))
})
log.Fatal(http.ListenAndServe(":8080", nil))
```

示例：

```
// HTTP/server/server.go

// HTTP server端
```

```
package main

import (
    "fmt"
    "net/http"
)

func main() {
    //http://127.0.0.1:8000/go
    // 单独写回调函数
    http.HandleFunc("/go", myHandler)
    //http.HandleFunc("/ungo",myHandler2 )
    // addr：监听的地址
    // handler：回调函数
    http.ListenAndServe("127.0.0.1:8000", nil)
}

// handler函数
func myHandler(w http.ResponseWriter, r *http.Request) {
    fmt.Println(r.RemoteAddr, "连接成功")
    // 请求方式：GET POST DELETE PUT UPDATE
    fmt.Println("method:", r.Method)
    // /go
    fmt.Println("url:", r.URL.Path)
    fmt.Println("header:", r.Header)
    fmt.Println("body:", r.Body)
    // 回复
    w.Write([]byte("我爱包子！！！"))
}
```