Go 语言实现 http 共享(总结)

go 语言入门简单，可要写出好的 go 代码得需要下些功夫。前两天刚把 http 文件共享的升级过（http://www.cnblogs.com/MikeZhang/archive/2012/08/06/httpShareGolang20120805.html ），现在经大牛指点完全用 http 实现，感觉爽快多了。

一个简单的 http 服务器代码

```
package main

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

// hello world, the web server
func HelloServer(w http.ResponseWriter, req *http.Request) {
        io.WriteString(w, "hello, world!\n")
}

func main() {
        http.HandleFunc("/hello", HelloServer)
        err := http.ListenAndServe(":12345", nil)
        if err != nil {
                log.Fatal("ListenAndServe: ", err)
        }
}
```

这里调用的是 http.HandleFunc 函数，这个函数声明如下：

```
func HandleFunc(pattern string, handler func(ResponseWriter, *Request))
```

这里用 HelloServer 实现 HandleFunc 函数的第二个参数。

一个简单的 http 服务器计数器程序代码

```
package main

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

type Counter struct {
        n int
}

func (ctr *Counter) ServeHTTP(c http.ResponseWriter, req *http.Request) {
```

```
        ctr.n++
        fmt.Fprintf(c, "counter = %d\n", ctr.n)
}

func main() {
        http.Handle("/counter", new(Counter))
        log.Fatal("ListenAndServe: ", http.ListenAndServe(":12345", nil))
}
```

这里调用的是 http.Handle 函数，这个函数声明如下：

```
func Handle(pattern string, handler Handler)
```

这里要说明的是：

几乎任何东西都可加以方法，几乎任何东西都可满足某界面，http 包定义的 Handler 界面就是这样，任何物件实现了 Handler 都可服务 HTTP 请求。

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

ResponseWriter 本身是个界面，它提供一些可访问的方法来返回客户的请求。这些方法包括标准的 Write 方法。因此 http.ResponseWriter 可用在 io.Writer 可以使用的地方。Request 是个结构，包含客户请求的一个解析过的表示。

所以这里只要给 Counter 实现 ServeHTTP 方法就可以服务 HTTP 请求了。

上面的两个例子是 http 服务器的两种方式，写出来主要是为了对比下。

共同点：

两种方式都实现了类似下面的函数

```
handler func(ResponseWriter, *Request)
```

不同点：
第一个例子的函数名字不固定，可以随便起，第二个例子中的函数名字只能是 ServeHTTP，其它的不行。

实现 http 文件共享

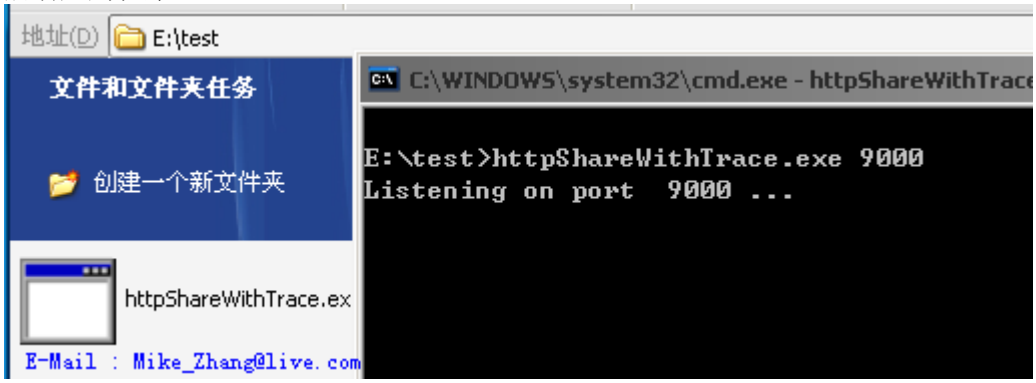
文件服务器要用到 http.FileServer 提供的 Handler，下面的代码就是基于第二种方式实现的。这里就不多解释了，不懂的参考上面的例子，比较下或许有帮助。

源代码如下：

```
package main

import (
    "net/http"
    "os"
    "strings"
    "log"
)
```

```
type TraceHandler struct {
    h http.Handler
}

func (r TraceHandler) ServeHTTP(w http.ResponseWriter, req *http.Request) {
        println("get",req.URL.Path," from ",req.RemoteAddr)
        r.h.ServeHTTP(w, req)
}

func main() {
        port := "8080"  //Default port
        if len(os.Args)>1 { port = strings.Join(os.Args[1:2],"")}
        h := http.FileServer(http.Dir("."))
        http.Handle("/", TraceHandler{h})
        println("Listening on port ",port,"...")
        log.Fatal("ListenAndServe: ", http.ListenAndServe(":"+port, nil))
}
```

启动应用程序：



Web 访问：