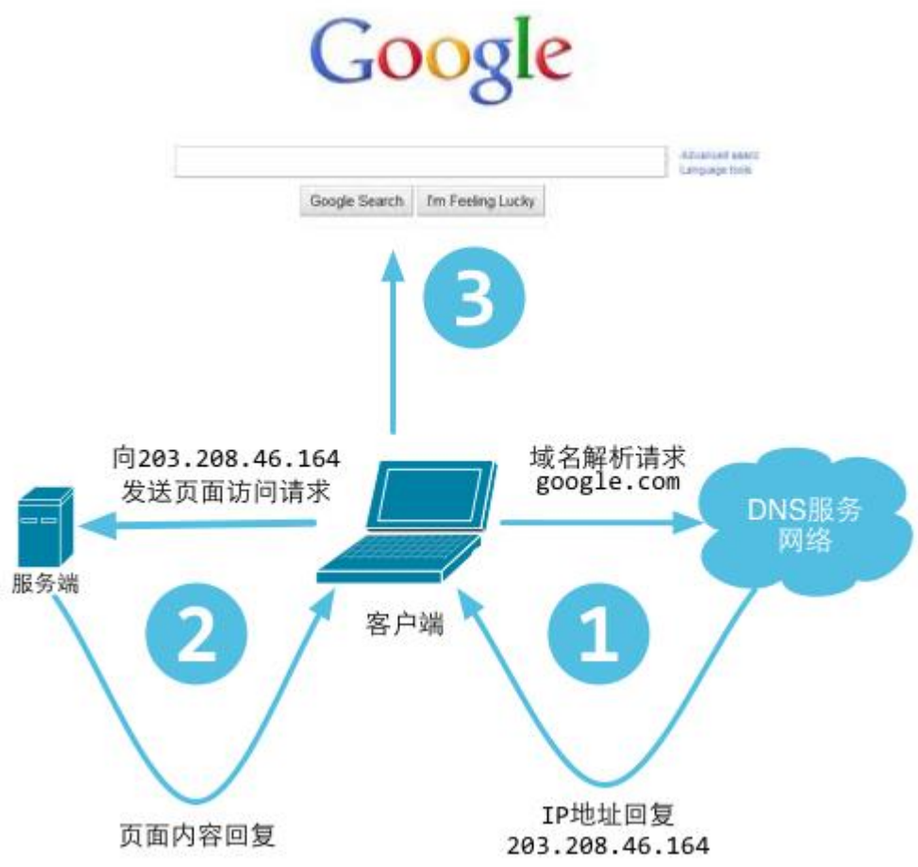
WEB编程

作者：少林之巅

目录

Web编程基础

1. Web工作方式



Web编程基础

2. HTTP协议详解

a. http 请求包体

```
GET /domains/example/ HTTP/1.1          //请求行: 请求方法 请求URI HTTP协议/协议版本
Host: www.iana.org                      //服务端的主机名
User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.4 (KHTML, like Gecko)
Chrome/22.0.1229.94 Safari/537.4                //浏览器信息
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8    //客户
端能接收的MIME
Accept-Encoding: gzip,deflate,sdch          //是否支持流压缩
Accept-Charset: UTF-8,*;q=0.5                //客户端字符编码集
//空行,用于分割请求头和消息体
//消息体,请求资源参数,例如POST传递的参数
```

Web编程基础

2. HTTP协议详解

b. http 响应包体

```
HTTP/1.1 200 OK                                    //状态行
Server: nginx/1.0.8                                //服务器使用的WEB软件名及版本
Date:Date: Tue, 30 Oct 2012 04:14:25 GMT           //发送时间
Content-Type: text/html                            //服务器发送信息的类型
Transfer-Encoding: chunked                         //表示发送HTTP包是分段发的
Connection: keep-alive                             //保持连接状态
Content-Length: 90                                 //主体内容长度
//空行 用来分割消息头和主体
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"... //消息体
```

3. http Keep-Alive特性

A. Keep-Alive用来保持连接

B. Keep-Alive通过web服务器进行设置，保持的时间

4. Web程序开发

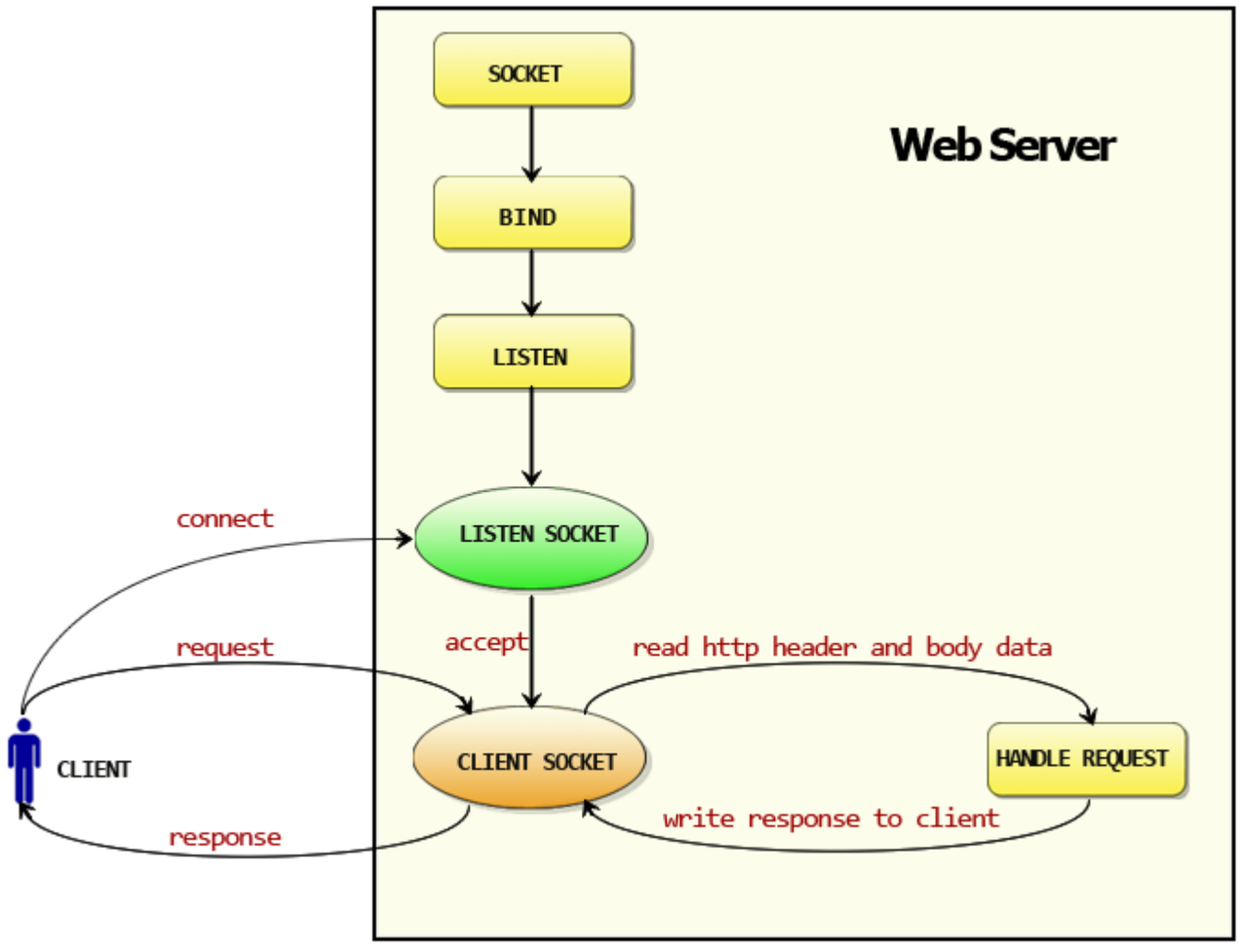
A. 标准包 “net/http”封装web服务相关功能

B. 使用简单、性能媲美nginx。

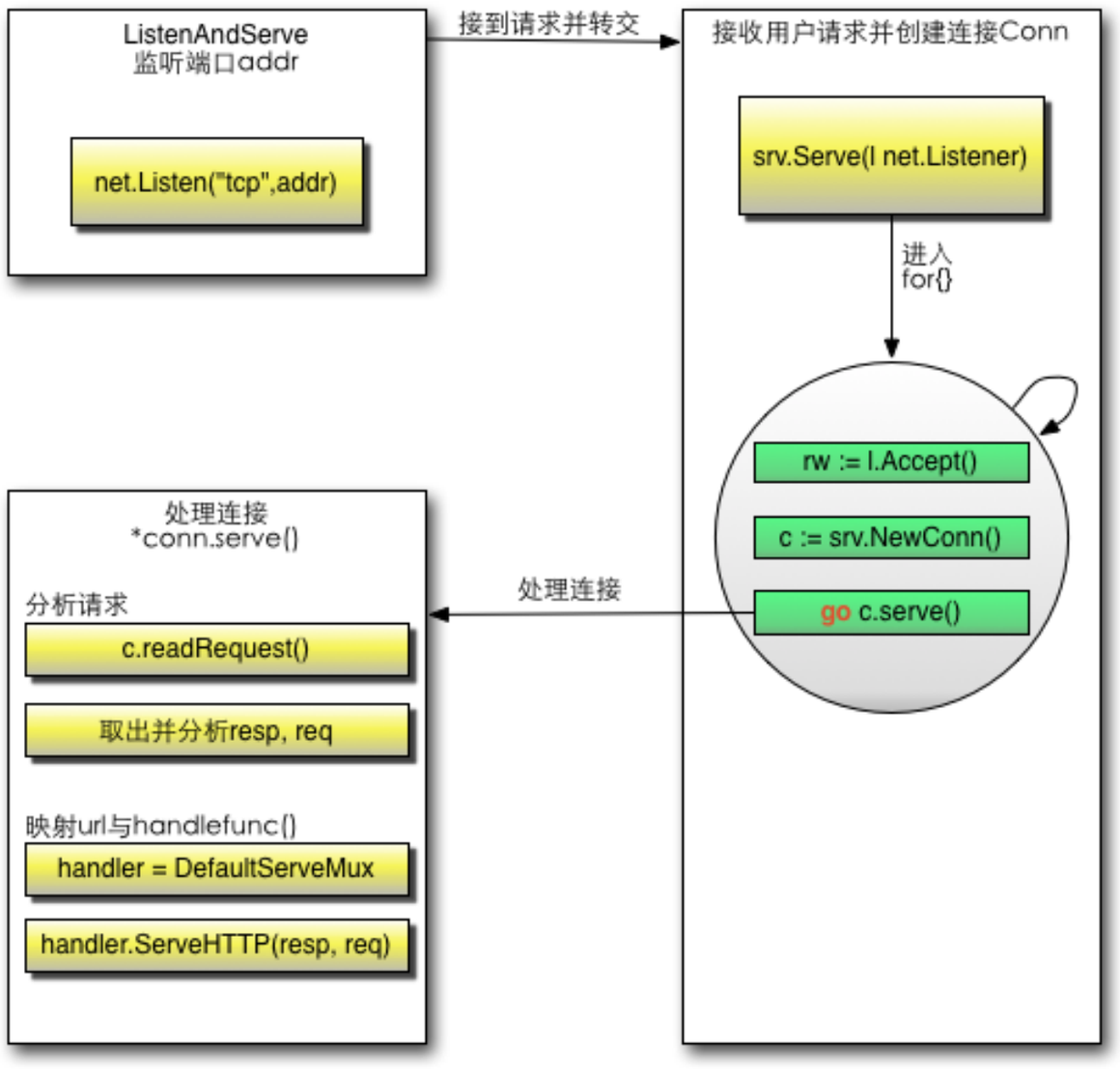
Web程序开发

```
package main

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

func sayhelloName(w http.ResponseWriter, r *http.Request) {
	r.ParseForm()  //解析参数，默认是不会解析的
	fmt.Println(r.Form)  //这些信息是输出到服务器端的打印信息
	fmt.Println("path", r.URL.Path)
	fmt.Println("scheme", r.URL.Scheme)
	fmt.Println(r.Form["url_long"])
	for k, v := range r.Form {
		fmt.Println("key:", k)
		fmt.Println("val:", strings.Join(v, ""))
	}
	fmt.Fprintf(w, "Hello world!") //这个写入到w的是输出到客户端的
}

func main() {
	http.HandleFunc("/", sayhelloName) //设置访问的路由
	err := http.ListenAndServe(":9090", nil) //设置监听的端口
	if err != nil {
		log.Fatal("ListenAndServe: ", err)
	}
}
```

5. Golang web服务工作方式



5. Golang web服务工作方式



6. Web表单

A. 通过<form> </form>括起来的区域，允许用户提交数据。

B. Go对于表单处理非常方便

Web表单

7. Html代码

```
<html>
<head>
<title></title>
</head>
<body>
<form action="/login" method="post">
     用户名:<input type="text" name="username">
     密码:<input type="password" name="password">
     <input type="submit" value="登录">
</form>
</body>
</html>
```

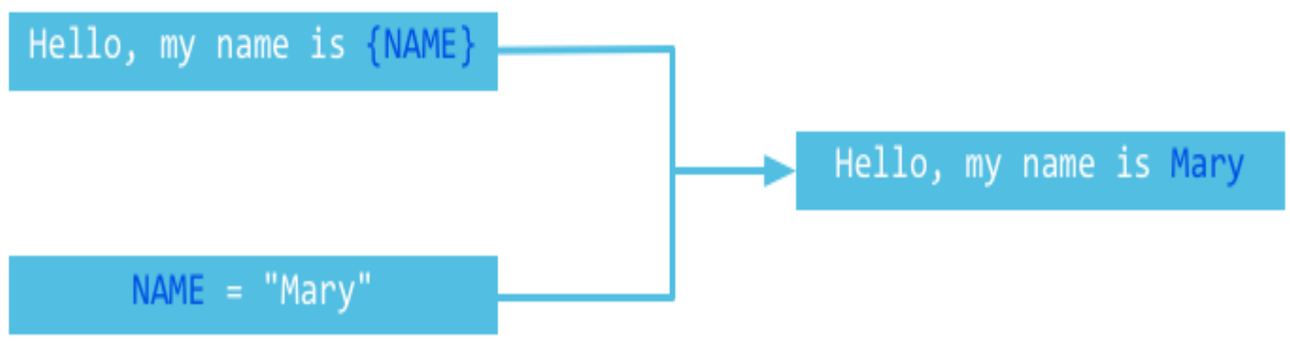
8. Go代码

```
package main

import (
	"fmt"
	"html/template"
	"log"
	"net/http"
	"strings"
)
func login(w http.ResponseWriter, r *http.Request) {
	fmt.Println("method:", r.Method) //获取请求的方法
	if r.Method == "GET" {
		t, _ := template.ParseFiles("login.gtpl")
		log.Println(t.Execute(w, nil))
	} else {
		//请求的是登录数据，那么执行登录的逻辑判断
		fmt.Println("username:", r.Form["username"])
		fmt.Println("password:", r.Form["password"])
	}
}
func main() {
	http.HandleFunc("/login", login)         //设置访问的路由
	err := http.ListenAndServe(":9090", nil) //设置监听的端口
	if err != nil {
		log.Fatal("ListenAndServe: ", err)
	}
}
```

9. 模板替换

A. {{}}来包含需要在渲染时被替换的字段，{{.}}表示当前的对象。

B. 通过{{.FieldName}}访问对象的属性。



9. 模板替换

```
package main

import (
        "fmt"
        "os"
        "text/template"
)

type Person struct {
        Name string
        age  string
}

func main() {
        t, err := template.ParseFiles("./index.html")
        if err != nil {
                fmt.Println("parse file err:", err)
                return
        }
        p := Person{Name: "Mary", age: "31"}
        if err := t.Execute(os.Stdout, p); err != nil {
                fmt.Println("There was an error:", err.Error())
        }
}
```

Web模板

10.If判断

```
<html>
	<head>
	</head>
	<body>
		{{if gt .Age 18}}
		<p>hello, old man, {{.Name}}</p>
		{{else}}
		<p>hello,young man, {{.Name}}</p>
		{{end}}
	</body>
</html>
```

- not 非

{{if not .condition}}
{{end}}

- and 与

{{if and .condition1 .condition2}}
{{end}}

- or 或

{{if or .condition1 .condition2}}
{{end}}

- eq 等于

{{if eq .var1 .var2}}
{{end}}

- ne 不等于

{{if ne .var1 .var2}}
{{end}}

- lt 小于 (less than)

{{if lt .var1 .var2}}
{{end}}

- le 小于等于

{{if le .var1 .var2}}
{{end}}

- gt 大于

{{if gt .var1 .var2}}
{{end}}

- ge 大于等于

{{if ge .var1 .var2}}
{{end}}

11. with 语法

```
<html>
    <head>
    </head>
    <body>
        {{with .Name}}
        <p>hello, old man, {{.}}</p>
        {{end}}
    </body>
</html>
```

Web模板

12.循环

```
<html>
    <head>
    </head>
    <body>
        {{range .}}
            {{if gt .Age 18}}
            <p>hello, old man, {{.Name}}</p>
            {{else}}
            <p>hello,young man, {{.Name}}</p>
            {{end}}
        {{end}}
    </body>
</html>
```