Go培训第10天

tony

Outline

1. http编程

2. mysql使用

3. 课后作业

http编程

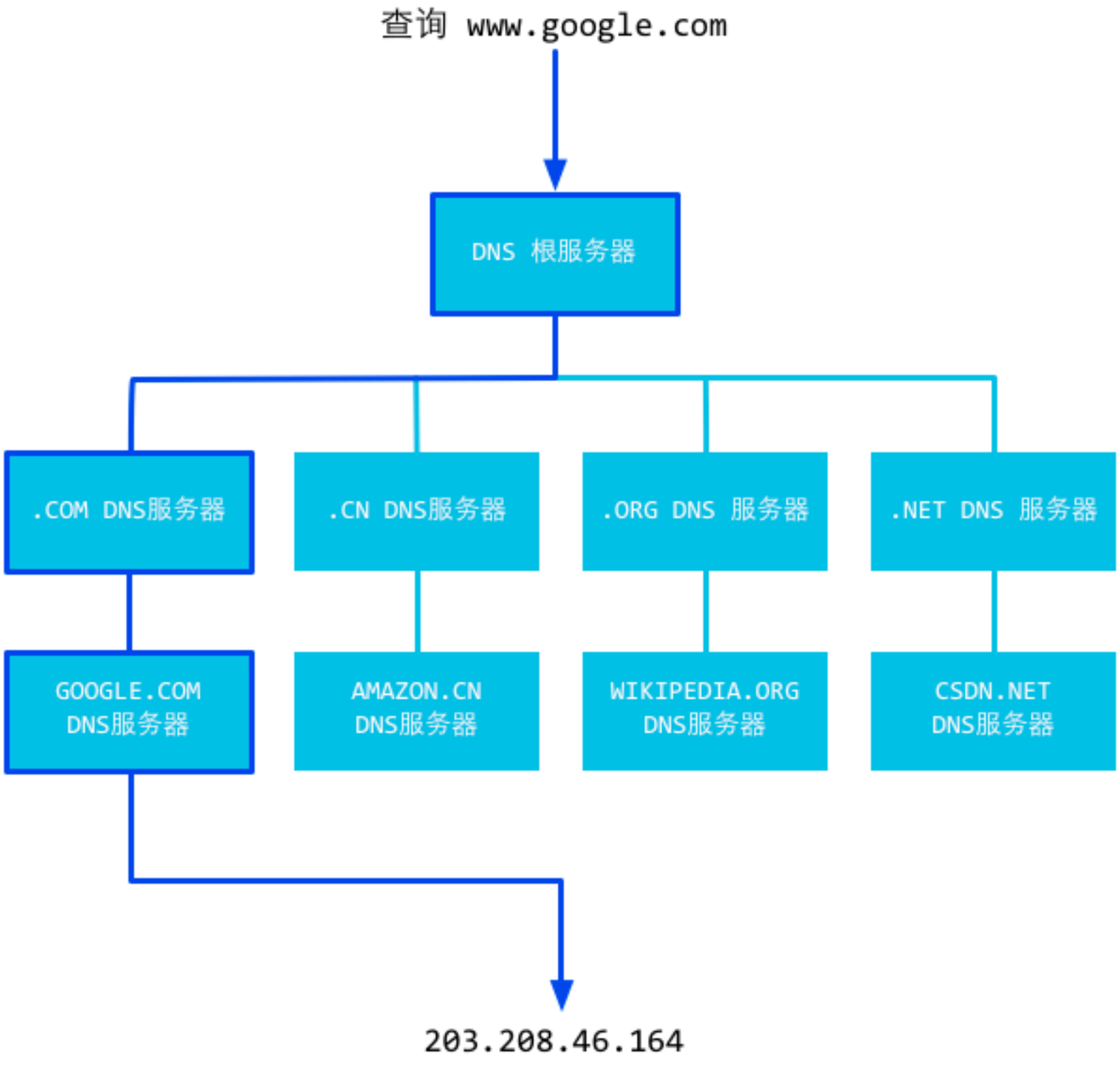
1. http编程

   a. Go原生支持http，import(“net/http”)

   b. Go的http服务性能和nginx比较接近

   c. 几行代码就可以实现一个web服务

http编程


查询 www.google.com
DNS 根服务器
.COM DNS服务器
.CN DNS服务器
.ORG DNS 服务器
.NET DNS 服务器
GOOGLE.COM
DNS服务器
AMAZON.CN
DNS服务器
WIKIPEDIA.ORG
DNS服务器
CSDN.NET
DNS服务器
203.208.46.164

http编程

1. http请求包

```
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/
    //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mine
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
```

2. http响应包

```
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编程

```
package main

import (
        "fmt"
        "net/http"
)

func Hello(w http.ResponseWriter, r *http.Request) {
        fmt.Println("handle hello")
        fmt.Fprintf(w, "hello ")
}

func main() {
        http.HandleFunc("/", Hello)
        err := http.ListenAndServe("0.0.0.0:8880", nil)
        if err != nil {
                fmt.Println("http listen failed")
        }
}
```

4. http client

```
package main

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

func main() {
        res, err := http.Get("https://www.baidu.com/")
        if err != nil {
                fmt.Println("get err:", err)
                return
        }

        data, err := ioutil.ReadAll(res.Body)
        if err != nil {
                fmt.Println("get data err:", err)
                return
        }

        fmt.Println(string(data))
}
```

5. http常见请求方法

1）**Get请求**

2）**Post请求**

3）Put请求

4）Delete请求

5）Head请求

6. head请求实例

```
package main

import (
	"fmt"
	"net/http"
)

var url = []string{
	"http://www.baidu.com",
	"http://google.com",
	"http://taobao.com",
}

func main() {

	for _, v := range url {
		resp, err := http.Head(v)
		if err != nil {
			fmt.Printf("head %s failed, err:%v\n", v, err)
			continue
		}

		fmt.Printf("head succ, status:%v\n", resp.Status)
	}
}
```

7. head请求实例

```
package main

import (
	"fmt"
	"net/http"
)

var url = []string{
	"http://www.baidu.com",
	"http://google.com",
	"http://taobao.com",
}

func main() {

	for _, v := range url {
		resp, err := http.Head(v)
		if err != nil {
			fmt.Printf("head %s failed, err:%v\n", v, err)
			continue
		}

		fmt.Printf("head succ, status:%v\n", resp.Status)
	}
}
```

8. http 常见状态码

http.StatusContinue = 100
http.StatusOK = 200
http.StatusFound = 302
http.StatusBadRequest = 400
http.StatusUnauthorized = 401
http.StatusForbidden = 403
http.StatusNotFound = 404
http.StatusInternalServerError = 500

9. 表单处理

```
package main
import (
        "io"
        "net/http"
)

const form = `<html><body><form action="#" method="post" name="bar">
                    <input type="text" name="in"/>
                    <input type="text" name="in"/>
                     <input type="submit" value="Submit"/>
                </form></html></body>`

func SimpleServer(w http.ResponseWriter, request *http.Request) {
        io.WriteString(w, "<h1>hello, world</h1>")
}

func FormServer(w http.ResponseWriter, request *http.Request) {
        w.Header().Set("Content-Type", "text/html")
        switch request.Method {
        case "GET":
                io.WriteString(w, form)
        case "POST":
                request.ParseForm()
                io.WriteString(w, request.Form["in"][0])
                io.WriteString(w, "\n")
                io.WriteString(w, request.FormValue("in"))
        }
}
func main() {
        http.HandleFunc("/test1", SimpleServer)
        http.HandleFunc("/test2", FormServer)
        if err := http.ListenAndServe(":8088", nil); err != nil {
        }
}
```

10. panic处理

```
package main

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

const form = `<html><body><form action="#" method="post" name="bar">
                    <input type="text" name="in"/>
                    <input type="text" name="in"/>
                     <input type="submit" value="Submit"/>
                </form></html></body>`

//代码省略
func main() {
        http.HandleFunc("/test1", logPanics(SimpleServer))
        http.HandleFunc("/test2", logPanics(FormServer))
        if err := http.ListenAndServe(":8088", nil); err != nil {
        }
}

func logPanics(handle http.HandlerFunc) http.HandlerFunc {
        return func(writer http.ResponseWriter, request *http.Request) {
                defer func() {
                        if x := recover(); x != nil {
                                log.Printf("[%v] caught panic: %v", request.RemoteAddr, x)
                        }
                }()
                handle(writer, request)
        }
}
```

11. 模板

1）替换 {{.字段名}}

```
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())
	}
}
```

12. 模板

1） 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>
```

13. 模板

2）if常见操作符

- 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}}

14. 模板

3）{{.}}

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

15. 模板

4）{{with .Var}}

5）{{end}}

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

16. 模板

1）循环

{{range.}}

{{end }}

```
<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>
```

2. mysql编程

a. 新建测试表

```
CREATE TABLE person (
    user_id int primary key auto_increment,
    username varchar(260),
    sex varchar(260),
    email varchar(260)
);
```

```
CREATE TABLE place (
    country varchar(200),
    city varchar(200),
    telcode int
)
```

Mysql

2. mysql编程

b. 链接mysql

```
database, err := sqlx.Open("mysql", "root:@tcp(127.0.0.1:3306)/test")
```

Mysql

2. mysql编程

b. insert操作

```
r, err := Db.Exec("insert into person(username, sex, email)values(?, ?, ?)", "stu001", "man", "stu01@qq.com")
```

```
package main

import (
	"fmt"
	_ "github.com/go-sql-driver/mysql"
	"github.com/jmoiron/sqlx"
)

type Person struct {
	UserId   int    `db:"user_id"`
	Username string `db:"username"`
	Sex      string `db:"sex"`
	Email    string `db:"email"`
}

type Place struct {
	Country string `db:"country"`
	City    string `db:"city"`
	TelCode int    `db:"telcode"`
}

var Db *sqlx.DB
func init() {
	database, err := sqlx.Open("mysql", "root:@tcp(127.0.0.1:3306)/test")
	if err != nil {
		fmt.Println("open mysql failed,", err)
		return
	}
	Db = database
}

func main() {
	r, err := Db.Exec("insert into person(username, sex, email)values(?, ?, ?)", "stu001", "man", "stu01@qq.com")
	if err != nil {
		fmt.Println("exec failed, ", err)
		return
	}
	id, err := r.LastInsertId()
	if err != nil {
		fmt.Println("exec failed, ", err)
		return
	}

	fmt.Println("insert succ:", id)
}
```

2. mysql编程

d. Select 操作

```
err := Db.Select(&person, "select user_id, username, sex, email from person where user_id=?", 1)
```

```
package main

import (
	"fmt"
	_ "github.com/go-sql-driver/mysql"
	"github.com/jmoiron/sqlx"
)

type Person struct {
	UserId   int    `db:"user_id"`
	Username string `db:"username"`
	Sex      string `db:"sex"`
	Email    string `db:"email"`
}

type Place struct {
	Country string `db:"country"`
	City    string `db:"city"`
	TelCode int    `db:"telcode"`
}
var Db *sqlx.DB
func init() {

	database, err := sqlx.Open("mysql", "root:@tcp(127.0.0.1:3306)/test")
	if err != nil {
		fmt.Println("open mysql failed,", err)
		return
	}

	Db = database
}

func main() {

	var person []Person
	err := Db.Select(&person, "select user_id, username, sex, email from person where user_id=?", 1)
	if err != nil {
		fmt.Println("exec failed, ", err)
		return
	}

	fmt.Println("select succ:", person)
}
```

2. mysql编程

b. update操作

```
_, err := Db.Exec("update person set username=? where user_id=?", "stu0001", 1)
```

```
package main

import (
	"fmt"
	_ "github.com/go-sql-driver/mysql"
	"github.com/jmoiron/sqlx"
)

type Person struct {
	UserId   int    `db:"user_id"`
	Username string `db:"username"`
	Sex      string `db:"sex"`
	Email    string `db:"email"`
}

type Place struct {
	Country string `db:"country"`
	City    string `db:"city"`
	TelCode int    `db:"telcode"`
}
```

```
var Db *sqlx.DB

func init() {

	database, err := sqlx.Open("mysql", "root:@tcp(127.0.0.1:3306)/test")
	if err != nil {
		fmt.Println("open mysql failed,", err)
		return
	}

	Db = database
}

func main() {

	_, err := Db.Exec("update person set username=? where user_id=?", "stu0001", 1)
	if err != nil {
		fmt.Println("exec failed, ", err)
		return
	}

}
```

2. mysql编程

d. Delete 操作

```
_, err := Db.Exec("delete from person where user_id=?", 1)
```

```
package main

import (
	"fmt"
	_ "github.com/go-sql-driver/mysql"
	"github.com/jmoiron/sqlx"
)

type Person struct {
	UserId   int    `db:"user_id"`
	Username string `db:"username"`
	Sex      string `db:"sex"`
	Email    string `db:"email"`
}

type Place struct {
	Country string `db:"country"`
	City    string `db:"city"`
	TelCode int    `db:"telcode"`
}

var Db *sqlx.DB

func init() {

	database, err := sqlx.Open("mysql", "root:@tcp(127.0.0.1:3306)/test")
	if err != nil {
		fmt.Println("open mysql failed,", err)
		return
	}

	Db = database
}

func main() {

	_, err := Db.Exec("delete from person where user_id=?", 1)
	if err != nil {
		fmt.Println("exec failed, ", err)
		return
	}

	fmt.Println("delete succ")
}
```

课后作业

1. 修改图书管理系统，把数据存储部分切换到mysql中。