连接数据库

下载驱动

终端输入 Plain Text

```
go get -u github.com/go-sql-driver/mysql
```

连接代码

连接类型：MySQL

用户名：root

密码：wwy040609

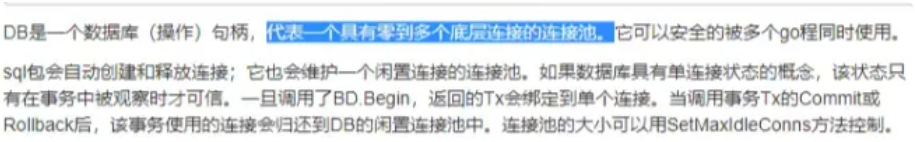
连接代码 Go

```go
package main

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

func main() {
    //连接数据库
    dsn := "root:wwy040609@tcp(127.0.0.1:3306)/mydb1"
    //连接数据库
    db, err := sql.Open("mysql", dsn)
    if err != nil {
        fmt.Printf("Open %s failed,err:%v\n", dsn, err)
        return//不会校验用户名和密码
    }
    err = db.Ping()
    if err != nil {
        //用于校验用户名和密码
        fmt.Printf("Open %s failed,err:%v\n", dsn, err)
        return
    }
    fmt.Println("连接成功")
}
```

mysql的Open函数返回

*DB：一个数据库的连接池

err：如果连接成功返回nil，否则返回连接错误类型。



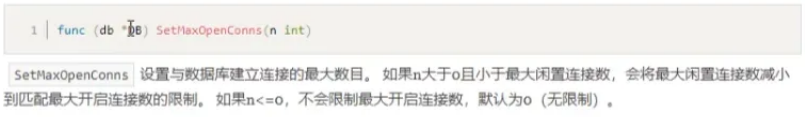
初始化数据库

```go
var database *sql.DB
func initDB() error {
    //连接数据库
    dsn := "root:wwy040609@tcp(127.0.0.1:3306)/mydb1"
    //连接数据库
    var err error
    database, err = sql.Open("mysql", dsn)
    if err != nil {
        fmt.Printf("Open %s failed,err:%v\n", dsn, err)
        return err
    }
    err = database.Ping()
    if err != nil {
        fmt.Printf("Open %s failed,err:%v\n", dsn, err)
        return err
    }
    return nil
}
```

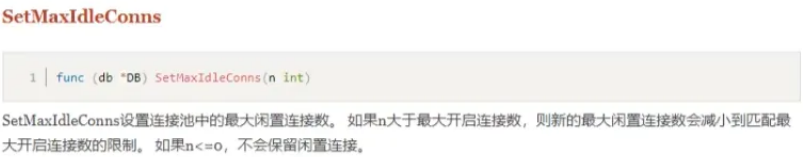
与主函数中直接连接数据库不同，如果和主函数中连接相同的话，声明的database作为连接池变量是局部变量，在主函数调用时是空指针。

连接池的连接数量

SetMaxOpenConns（控制连接查询的次数）



SetMaxIdelConns（设置连接池中最大空闲连接数）



数据库操作

查询单条数据

```go
func query() {
    var ID int
    fmt.Scanf("%d", &ID)
    sqlStr := "select * from stu where id=?"
    rowObj := database.QueryRow(sqlStr, ID)
    var stu student
    rowObj.Scan(&stu.name, &stu.password, &stu.age, &stu.ID)
    //Scan函数执行结束后会自动关闭数据库
    fmt.Println(stu)
}
```

查询多条数据

```go
func all_query() {
    sqlStr := "select * from stu where id > 0"
    rowObj, err := database.Query(sqlStr)
    if err != nil {
        fmt.Printf("Query %s failed,err:%v\n", sqlStr, err)
        return
    }
    defer rowObj.Close()
    for rowObj.Next() {
        var stu student
        err := rowObj.Scan(&stu.name, &stu.password, &stu.age, &stu.ID)
        if err != nil {
            fmt.Printf("scan failed,err:%v\n", err)
            return
        }
        fmt.Println(stu)
    }
}
```

插入数据和更新数据

插入数据 Go

```go
func insert() {
    var name string
    var age int
    var Id int
    var password string
    fmt.Println("请一次输入年龄，密码，ID号，姓名")
    fmt.Scanf("%d", &age)
    fmt.Scanf("%s", &password)
    fmt.Scanf("%d", &Id)
    fmt.Scanf("%s", &name)
    //1.先写sql语句
    sqlStr := "insert into stu(password,name,age,ID) values(?,?,?,?)"
    //2.exec
    ret, err := database.Exec(sqlStr, password, name, age, Id)
    if err != nil {
        fmt.Printf("insert failed,err:%v\n", err)
        return
    }
    //如果是插入数据的操作，能拿到插入数据的ID值
    id, err := ret.LastInsertId()
    if err != nil {
        fmt.Printf("get id failed,err:%v\n", err)
    }
    fmt.Printf("insert success, id:%d\n", id)
}
```