ORM-SQL

SQLSQL**SQLSQL**SQLSQLORMSQL

Content Menu

CatchSQL

gdb.CatchSQLSQL

```
// CatchSQL catches and returns all sql statements that are EXECUTED in
given closure function.
// Be caution that, all the following sql statements should use the
context object passing by function `f`.
func CatchSQL(ctx context.Context, f func(ctx context.Context) error)
(sqlArray []string, err error)
```

SQLSQL[]stringSQLctxSQL

user.sql

```
CREATE TABLE `user` (
    `id`          int(10) unsigned NOT NULL AUTO_INCREMENT,
    `passport`    varchar(45) NULL,
    `password`    char(32) NULL,
    `nickname`    varchar(45) NULL,
    `create_time` timestamp(6) NULL,
    PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

`main.go`

```
package main

import (
        "context"

        _ "github.com/gogf/gf/contrib/drivers/mysql/v2"
        "github.com/gogf/gf/v2/database/gdb"
        "github.com/gogf/gf/v2/frame/g"
        "github.com/gogf/gf/v2/os/gctx"
        "github.com/gogf/gf/v2/os/gtime"
)

type User struct {
        Id         int
        Passport   string
        Password   string
        Nickname   string
        CreateTime *gtime.Time
}

func initUser(ctx context.Context) error {
        _, err := g.Model("user").Ctx(ctx).Data(User{
                Id:       1,
                Passport: "john",
                Password: "12345678",
                Nickname: "John",
        }).Insert()
        return err
}

func main() {
        var ctx = gctx.New()
        sqlArray, err := gdb.CatchSQL(ctx, func(ctx context.Context) error
{
                return initUser(ctx)
        })
        if err != nil {
                panic(err)
        }
        g.Dump(sqlArray)
}
```

```
[
    "SHOW FULL COLUMNS FROM `user`",
    "INSERT INTO `user`(`id`,`passport`,`password`,`nickname`,`created_at`,
`updated_at`) VALUES(1,'john','12345678','John','2023-12-19 21:43:
57','2023-12-19 21:43:57') ",
]
```

ToSQL

gdb.ToSQLSQLSQL

```
// ToSQL formats and returns the last one of sql statements in given
closure function
// WITHOUT TRULY EXECUTING IT.
// Be caution that, all the following sql statements should use the
context object passing by function `f`.
func ToSQL(ctx context.Context, f func(ctx context.Context) error) (sql
string, err error)
```

SQLSQLSQLstringSQLctxSQL

user.sql

```
CREATE TABLE `user` (
    `id`          int(10) unsigned NOT NULL AUTO_INCREMENT,
    `passport`    varchar(45) NULL,
    `password`    char(32) NULL,
    `nickname`    varchar(45) NULL,
    `create_time` timestamp(6) NULL,
    PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

main.go

```
package main

import (
        "context"

        _ "github.com/gogf/gf/contrib/drivers/mysql/v2"
        "github.com/gogf/gf/v2/database/gdb"
        "github.com/gogf/gf/v2/frame/g"
        "github.com/gogf/gf/v2/os/gctx"
        "github.com/gogf/gf/v2/os/gtime"
)

type User struct {
        Id         int
        Passport   string
        Password   string
        Nickname   string
        CreateTime *gtime.Time
}

func initUser(ctx context.Context) error {
        _, err := g.Model("user").Ctx(ctx).Data(User{
                Id:       1,
                Passport: "john",
                Password: "12345678",
                Nickname: "John",
        }).Insert()
        return err
}

func main() {
        var ctx = gctx.New()
        sql, err := gdb.ToSQL(ctx, func(ctx context.Context) error {
                return initUser(ctx)
        })
        if err != nil {
                panic(err)
        }
        g.Dump(sql)
}
```

```
"INSERT INTO `user`(`id`,`passport`,`password`,`nickname`,`created_at`,
`updated_at`) VALUES(1,'john','12345678','John','2023-12-19 21:49:
21','2023-12-19 21:49:21') "
```