do

dodonil

doAPI

```
CREATE TABLE `user`(
    `id`        int(10) unsigned NOT NULL AUTO_INCREMENT COMMENT 'ID',
    `passport`  varchar(32) NOT NULL COMMENT '',
    `password`  varchar(32) NOT NULL COMMENT '',
    `nickname`  varchar(32) NOT NULL COMMENT '',
    `status`    varchar(32) NOT NULL COMMENT '',
    `brief`     varchar(512) NOT NULL COMMENT '',
    `create_at` datetime DEFAULT NULL COMMENT '',
    `update_at` datetime DEFAULT NULL COMMENT '',
    PRIMARY KEY (`id`),
    UNIQUE KEY `uniq_passport` (`passport`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
type Status string

const (
        StatusEnabled  = "enabled"
        StatusDisabled = "disabled"
)
```

gf gen daodo

```
type User struct {
        g.Meta    `table:"uer" orm:"do:true"`
        Id        interface{}
        Passport  interface{}
        Password  interface{}
        Nickname  interface{}
        Status    interface{}
        Brief     interface{}
        CreatedAt interface{}
        UpdatedAt interface{}
}
```

API

APIAPI

```
type UpdateReq struct {
        g.Meta   `path:"/user/{Id}" method:"post" summary:""`
        Passport string  `v:"required" dc:""`
        Password *string `dc:""`
        Nickname *string `dc:""`
        Status   *Status `dc:""`
        Brief    *string `dc:""`
}
```

donildonildonil

```
func (c *Controller) Update(ctx context.Context, req *v1.UpdateReq) (res *v1.UpdateRes, err error) {
        _, err = dao.User.Ctx(ctx).Data(do.User{
                Password: req.Password,
                Nickname: req.Nickname,
                Status:   req.Status,
                Brief:    req.Brief,
        }).Where(do.User{
                Passport: req.Passport,
        }).Update()
        return
}
```