依赖管理和MYSQL开发

作者：少林之巅

目录

依赖管理介绍

1. 简介

   A. 所有的第三方包都放在$GOPATH的src目录下。

   B. 如果不同程序依赖的版本不一样，怎么管理

   C. 每个程序依赖的包，没有版本号的概念。

Godep使用简介

2. godep安装

   a. 安装方法： go get github.com/tools/godep

   b. 输入godep 命令，出现帮助信息

Godep使用简介

3. Godep使用

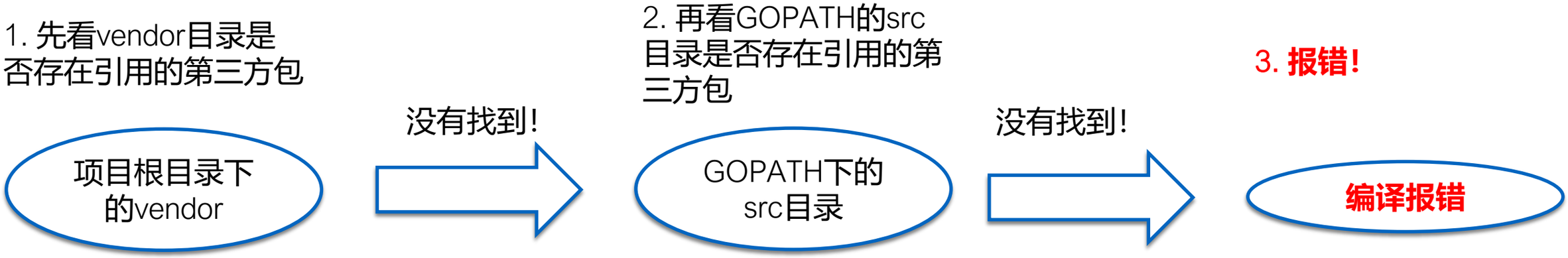
A. godep save， 把程序所有依赖的第三包信息保存起来

B. 生成Godep目录，保存第三包依赖的版本信息。

C. 生成vendor目录， 保存所有依赖的第三方包。

Vendor机制介绍

4. 控制包搜索路径的优先级， Go 1.6之后版本支持。



Godep使用简介

5. Godep开发流程

A. 保证程序能正常编译。

B. 执行godep save，保存当前所有第三方依赖的版本信息和代码

C. 提交Godeps目录和vendor目录到代码库。

D. 如果要更新依赖的版本，可以直接修改Godeps.json文件。

6. MYSQL常用引擎介绍

A. MyIASM引擎。

1. 不支持事务。
2. 不支持行锁。
3. 读性能比较好。

B. Innodb引擎。

1. 支持事务。
2. 支持行锁。
3. 整体性能比较好。

7. Golang中MYSQL驱动

A. https://github.com/go-sql-driver/mysql

B. Go本身不提供具体数据库驱动，只提供驱动接口和管理。

C. 各个数据库驱动需要第三方实现，并且注册到Go中的驱动管理中。

8. Mysql驱动，注册示例

GitHub, Inc. [US] | https://github.com/go-sql-driver/mysql/blob/master/driver.go

```
139                     maxap, err := mc.getSystemVar("max_allowed_packet")
140                     if err != nil {
141                             mc.Close()
142                             return nil, err
143                     }
144                     mc.maxAllowedPacket = stringToInt(maxap) - 1
145             }
146             if mc.maxAllowedPacket < maxPacketSize {
147                     mc.maxWriteSize = mc.maxAllowedPacket
148             }
149
150             // Handle DSN Params
151             err = mc.handleParams()
152             if err != nil {
153                     mc.Close()
154                     return nil, err
155             }
156
157             return mc, nil
158     }
159
160     func init() {
161             sql.Register("mysql", &MySQLDriver{})
162     }
```

9. 导入Mysql驱动

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

10. 连接数据库

```
func main() {
    dsn := "user:password@tcp(127.0.0.1:3306)/test"
    db, err := sql.Open("mysql",dsn)
    if err != nil {
        panic(err)
        return
    }
    defer db.Close()
}
```

11. 创建表

```
CREATE TABLE `user` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `name` varchar(20) DEFAULT ‘’,
  `age` int(11) DEFAULT '0’,
   PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT  CHARSET=utf8mb4;
```

MYSQL开发

12. sql查询

A. 单行查询， Db.QueryRow

B 多行查询， Db.Query

13. Mysql 插入、更新和删除

A. Db.Exec(query， args…interface{})(Result, error)

B. 插入的主键id获取，Result. LastInsertId()

MYSQL开发

14. Mysql预处理

A. 一般sql处理流程

1. 客户端拼接好sql语句
2. 客户端发送sql语句到mysql服务器
3. mysql服务器解析sql语句并执行，把执行结果发送给客户端

B. 预处理处理流程

1. 把sql分为两部分，命令部分和数据部分。
2. 首先把命令部分发送给mysql服务器，mysql进行sql预处理
3. 然后把数据部分发送给mysql服务器，mysql进行占位符替换
4. mysql服务器执行sql语句并返回结果给客户端

15. Mysql预处理优势

A. 同一条sql反复执行，性能会很高。

B. 避免sql注入问题

16. Mysql预处理实例

A. 查询操作，

1. Db.Prepare(sql string) (*sql.Stmt, error)

2. Stmt.Query()

B. 更新操作（插入、更新、delete），

1. Db.Prepare(sql string) (*sql.Stmt, error)

2. Stmt.Exec()

17. Mysql事务

A. 应用场景，

1. 同时更新，多个表。
2. 同时更新多行数据。

B. 事务的ACID

1. 原子性
2. 一致性
3. 隔离性
4. 持久性

18. Mysql事务实例

A. Db.Begin()，开启一个事务

B. Db.Commit()，提交一个事务

C. Db.Rollback()，回滚一个事务

19. sqlx库介绍与使用

A. 使用更简单

B. 支持多数据库，mysql、postgresql、oracle、sqlite

20. sqlx库介绍与使用

A. 查询， sqlx.DB.Get和sqlx.DB.Select

B. 更新、插入和删除，sqlx.DB.Exec

C. 事务，sqlx.DB.Begin()、sqlx.DB.Commit、sqlx.DB.rollback

21. sql注入分析

A. Select *from user where name = ‘%s’，构造name=“1 ' or 1 = 1 or ‘”

B. 构造name=123' and (select count(*) from user) > 10#

C. 构造name=123' union select *from user #

避免手动拼接sql， 使用占位符或预处理！