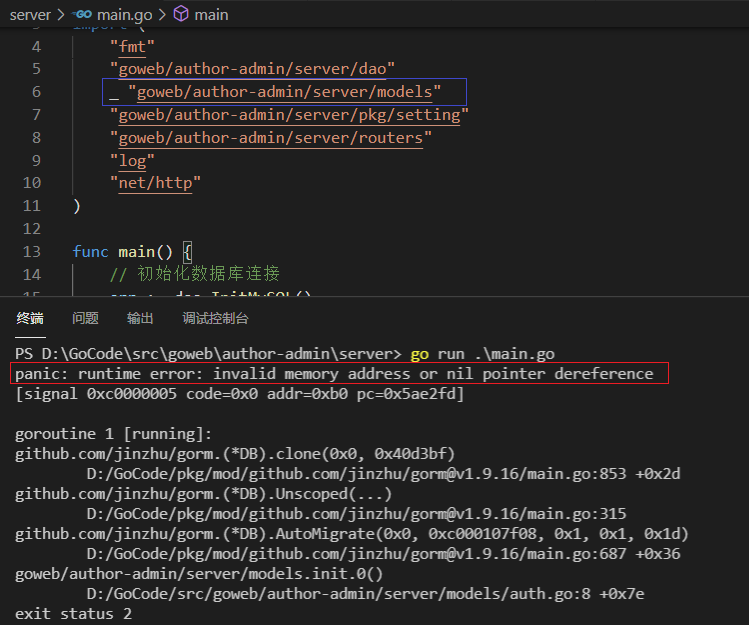
**1.错误发生场景**

我们开始了一个专家库系统author-admin的项目。在后台开发中，models包是ORM的实现；dao包定义与数据库的连接，利用了\*gorm.DB类型的包变量DB遇到一个与初始化相关的异常。

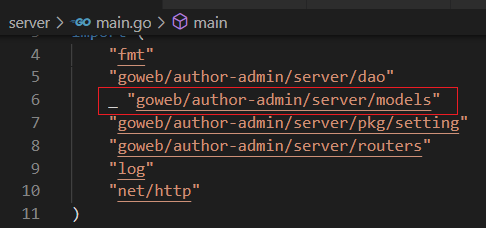
异常提示信息如下：



**2.解决问题**

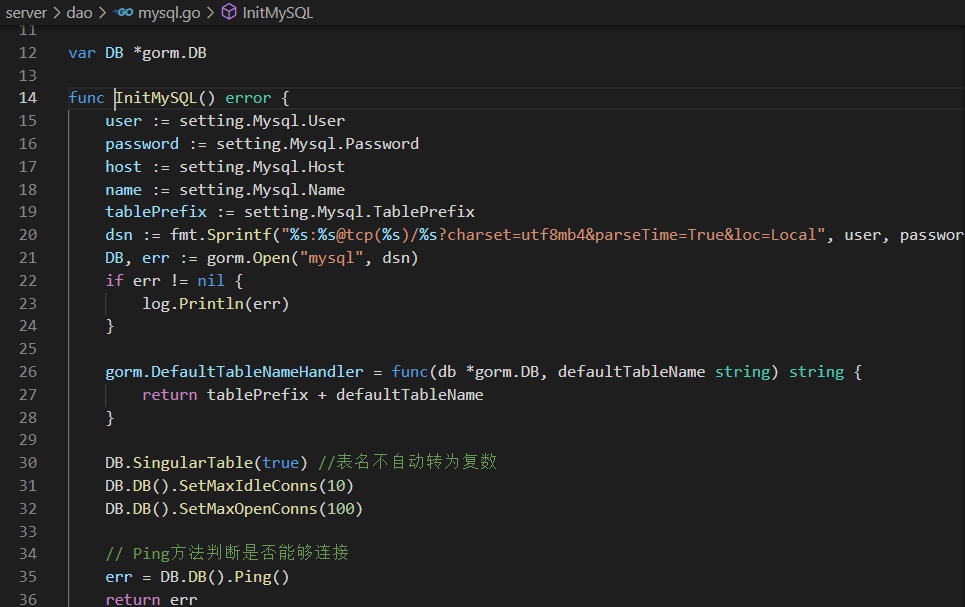
错误Runtime error: invalid memory address or nil pointer dereference，通常是因为使用了一个没有经过初始化的类型或者空指针变量。问题是我很奇怪怎么出来的这样一个错误，在精心安排了初始化之后。

在models包中，声明了大量结构体，它们与数据库表格关联起来。为了免去手动建表，我们用了\*gorm.DB类型的AutoMigrate方法。BTW，这一方法只会增加字段，不会删除和修改原字段。我们将AutoMigrage方法放在models包的init函数中，为了尽快完成“迁移”的初始化，在整个项目的main.go包中做了匿名导入：

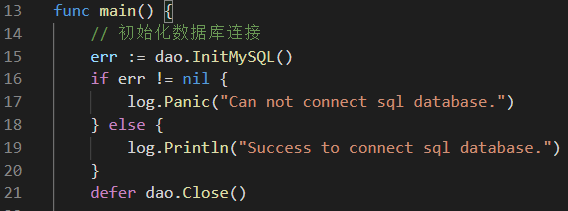


导入是会首先运行被导入包的init方法，利用匿名导入完成models包的初始化。BTW，models包有很多文件，每个文件里预期都会有init函数，这是Go语言允许的。

dao包的初始化没有利用init函数，而是实现了一个InitMySQL函数：



在main.go中调用做初始化：



这个好处是，可以让初始化函数有返回值，针对不同的返回值应用做出不同的反应。

问题恰恰就出现在这里。dao包声明的DB指针型全局变量，实在InitMySQl函数中初始化的，main.go包先导入了dao，后导入了models，运行了models的init函数；但此时dao包初始化还没有进行，该初始化是在main函数中，发生在包的导入之后，所以就会出现了使用没有初始化的指针类型变量的错误。

既然发现了问题所在，解决思路也就有了：

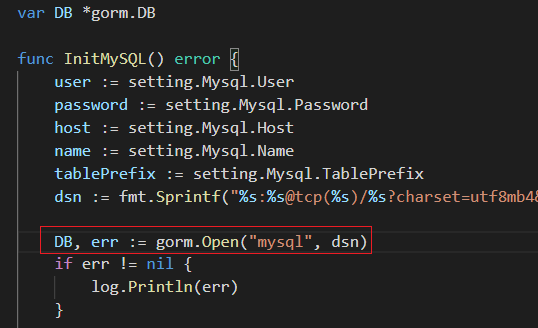
1）统一用init函数初始化。存在问题，跟导入的顺序有关，导入顺序IDE会给你自动按某种规律排序，很容易出现问题。

2）统一用自定义的InitXX来初始化。同样存在问题，与init函数不同，自定义的函数在同一个包中，如models包，只能出现一次。

3）在models包中，创建独立的init.go文件，统一初始化各个结构体的migration。这个思路似乎不错。

**3.最终**

当我按照上节方法3）做了修改，以为问题会迎刃而解的时候，发现异常仍然存在。通过设置多个断点，查看dao.DB变量的值，发现：



因为err前文未声明，所以此处不得不用:=符号赋值，于是出现了与我设想不同的一个特性，DB也视作了重新声明的局部变量，未给全局变量赋值。我的想当然导致了这个错误，改为下述语句最终解决问题：

var err error

DB, err = gorm.Open(“mysql”, dsn)