Go语言实现枚举的示例代码

作者：有故事的人 来源：范文网 www.wtabcd.cn/fanwen/

本文原地址：https://www.wtabcd.cn/fanwen/zuowen/510666cd4de84da4bd93462a972d11f1.html

在编程领域里，枚举用来表示只包含有限数量的固定值的类型，在开应急处置预案发中一般用于标识错误码或者状态机。拿一个实体对象的状态机来说，它通常与这个对象在数据库里对应记录的标识状态的字段值相对应。

在刚开始学编程的时候，你一定写过，至少见过直接使用魔术数字进行判断的代码。啥叫魔术数字呢，举个例子，要置顶一个文章的时候先判断文章是不是已发布状态。

```
if (article.state == 2) { // state 2 代表文章已发布}
```

假如我们的代码里没有注释，或者等我们项目的代码里充斥着这些魔术数字的判断的时候，你是不是会很头疼？
后来我就学会了把这些状态值定义成常量，并且也搞一个判断对象状态的方法单独封装这段逻辑。

```
public class articlestate {  public static final int draft = 1; //草稿  public sta
tic final int published = 2; //发布  public static final int deleted = 3; // 已
删除}public boolean checkarticlestate(int state) {  ... }
```

这种用法，肯定是比在程序里直接用魔术数字进行判断要强很多啦，至少看着不会很头疼，不会想骂**。

不过后来被当时带我的老大哥说这种也有缺点，上面这个 checkarticlestate
方法用来检查文章状态，本意是让调用者传入 articlestate
的三个静态常量之一，但由于没有类型上的约束，因此传入任意一个 int
值在语法上也是允许的，编译器也不会提出任何警告，用枚举更合适一些。
em~! 我不记得大学教 java 的那个学期老师讲过这玩意啊，莫非又是一个上课玩手机错过的知识点？所以使用枚举后我们的java代码变成了：

```
// 使用enum而非class声明public enum articlestate {  //要在enum里创建所有的枚举对象
 draft(1, "草稿");  published(2, "已发布");  deleted(3, "已删除")  // 自定义属
性  private int code;  private string text;  // 构造方法必须是private的  articlestate
(int code, string text) {  this.code = id;  this.text = name;  }}public b
oolean checkarticlestate(articlestate state) {  ... }
```

这样就能靠形参的枚举类型帮我们过滤掉非法的状态值，把整型值作为参数传给 checkarticlestate 方法时因为类型不匹配编译不过去，在写代码是编译器也能马上提示出来。

如果没有用过 java 的小伙伴也不用纠结篮球火台词，主要语法点我用注释标注出来了，大家应该都能看懂。后来这两年主要在用go做项目，我发现相似的问题 go 里存在，但是 go 并没有提供枚举类型，那怎么做到进行状态值的正确限制呢？如果还是用 int 型的常量肯定不行。比如：

const (draft int = 1 published = 2 deleted = 3)const (summer int = 1 autumn = 2 winter = 3 spring = 4)func main() { // 输出 true, 不会有任何编译错误 fmt.println(autumn == draft)}

比如上面定义了两肖邦是哪国人组 int 类型的常量，一类代表文章状态，一类代表季节的四季。这种方式拿文章状态与季节进行比较不会有任何编译上的错误。

答案在 go 内置库或者一些咱们都知道的开源库的代码里就能找到。比如看看 google.golang.org/grpc/codes 里的grpc 的错误码是怎么定义的，就能明白该怎么正确的实现枚举。

我们可以用 int 作为基础

type season intconst (summer season = 1 autumn = 2 winter = 3 spring = 4)

类型创建一个别名类型，go 里边是支持这个的

当然定义连续的常量值的时候 go 里边经常使用 iota，所以上面的定义还能进一步简化。

type season intconst (summer season = iota + 1 autumn winter spring)type article state intconst (draft int = iota + 1 published deleted)func checkarticlestate(state a rticlestate) bool { // ...}func main() { // 两个操作数类型不匹配，编译错误 fmt.println(autumn == draft) // 参数类型不匹配，但是因为 articlestate 底层的类型是 int 所以传递 int 的时候会发生隐式类型转换，所以不会报错 checkarticlestate(100)}

虽然这些状态值的底层的类型都是 int 值，但是现在进行两个不相干类型的枚举值比较，会造成编译错误，因为现在我们使用状态值的地方都有了类型限制。
不过函数 checkarticlestate 的参数类型设置成了益 articlestate 但是因为 articlestate 底层的类型是 int 。所以调用 checkarticlestate 时传递 int 类型的参数会发生隐式类型转换，不会造成编译报错，这块如果想解决，只能重新定义类型来实现了，可以参考stackoverflow上的这个答案

到此这篇关于go语言实现枚举的示例代码的文章就介绍到这了,更多相关go语言 枚举内容请搜索ww碳酸氢铵分解w.887551.com以前的文章或继续浏览下面的相关文章希望大家以后多多支持www.887551.com ！