What does an underscore and interface name after keyword var mean?

Asked 8 years ago Active 2 years, 3 months ago Viewed 14k times

79

27

From http://golang.org/src/pkg/database/sql/driver/types.go:

```go
type ValueConverter interface {
    // ConvertValue converts a value to a driver Value.
    ConvertValue(v interface{}) (Value, error)
}

var Bool boolType

type boolType struct{}

var _ ValueConverter = boolType{} // line 58

func (boolType) String() string { return "Bool" }

func (boolType) ConvertValue(src interface{}) (Value, error) {....}
```

I known that ValueConverter is an interface name. Line 58 seems to declare that boolType implement interface ValueConverter, but is that necessary? I deleted line 58 and the code works well.

syntax interface go underscores

edited Aug 15 '18 at 12:07
Flimzy
56.5k ●13 ●87 ●128

asked Nov 2 '12 at 11:25
dilfish
989 ●1 ●8 ●10

The blank identifier `_` can be used to strictly provide the keys in a struct too. See this for reference – Vallie Nov 18 '19 at 12:05

2 Answers

Active Oldest Votes

116

It provides a static (compile time) check that `boolType` satisfies the `ValueConverter` interface. The `_` used as a name of the variable tells the compiler to effectively discard the RHS value, but to type-check it and evaluate it if it has any side effects, but the anonymous variable per se doesn't take any process space.

It is a handy construct when developing and the method set of an interface and/or the methods implemented by a type are frequently changed. The construct serves as a guard against forgetting to match the method sets of a type and of an interface where the intent is to have them compatible. It effectively prevents to `go install` a broken (intermediate) version with such omission.

answered Nov 2 '12 at 11:46
zzzz
68.6k ●14 ●154 ●129

27

It seems like you are creating a dummy value of type `ValueConverter`, assigning a new `boolType` object to it and then discarding it (which is the meaning of the underscore in go, as in `for _, elt := range myRange { ...}` if you are not interested in the index of the enumeration).

My guess is that it simply correspond to a static check to ensure that the struct `boolType` does implement the `ValueConverter` interface. This way, when you change the implementation of `boolType`, the compiler will complain early if you broke the implementation of `ValueConverter` interface as it will be unable to cast your new `boolType` to this interface.

answered Nov 2 '12 at 11:47
val
7,205 ●25 ●30