ЛЕКЦИЯ 5

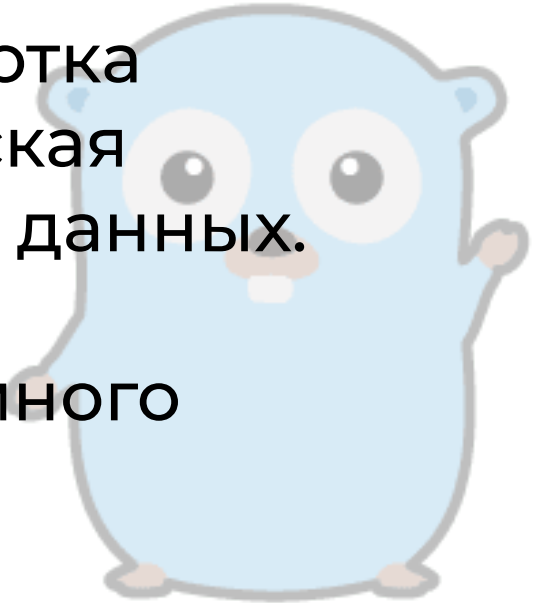
Рефлексия. Работа с JSON и XML

Рефлексия. Reflection

Определение:

Возможность интроспекции во время выполнения, то есть доступ и обработка значений любых типов и динамическая настройка на типы обрабатываемых данных.

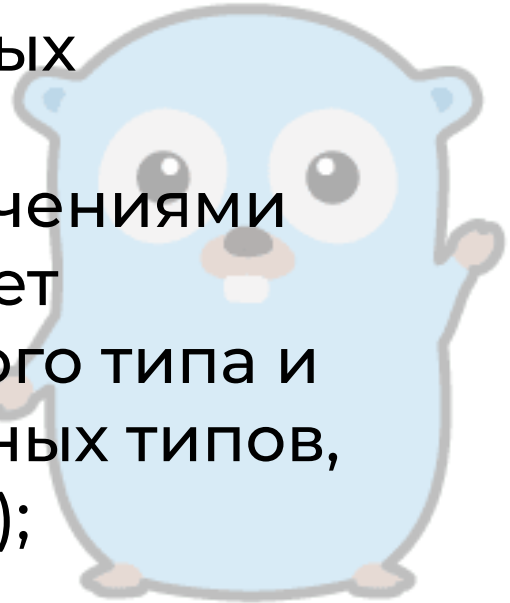
Реализуется в GO с помощью системного пакета "reflect".



Рефлексия. Reflection

Возможности пакета "reflect":

- Определить тип любого значения
- Сравнить на эквивалентность два любых значения
- Работать одним и тем же кодом со значениями любых типов (тип reflect.Value позволяет представить значение любого языкового типа и преобразовать его в один из стандартных типов, если такое преобразование возможно);



Рефлексия. Reflection

Возможности пакета "reflect":

- Изменять любые значения, если такое изменение в принципе возможно
- Исследовать типы, в частности, обращаться к полям структур, получать списки методов типов, их описания
- Вызывать произвольные функции и методы.

Рефлексия. Reflection

Чтобы создать переменную с динамическим типом, используются, так называемые, пустые интерфейсы.

Пример пустого интерфейса:

```
var SomeVar interface{}
```

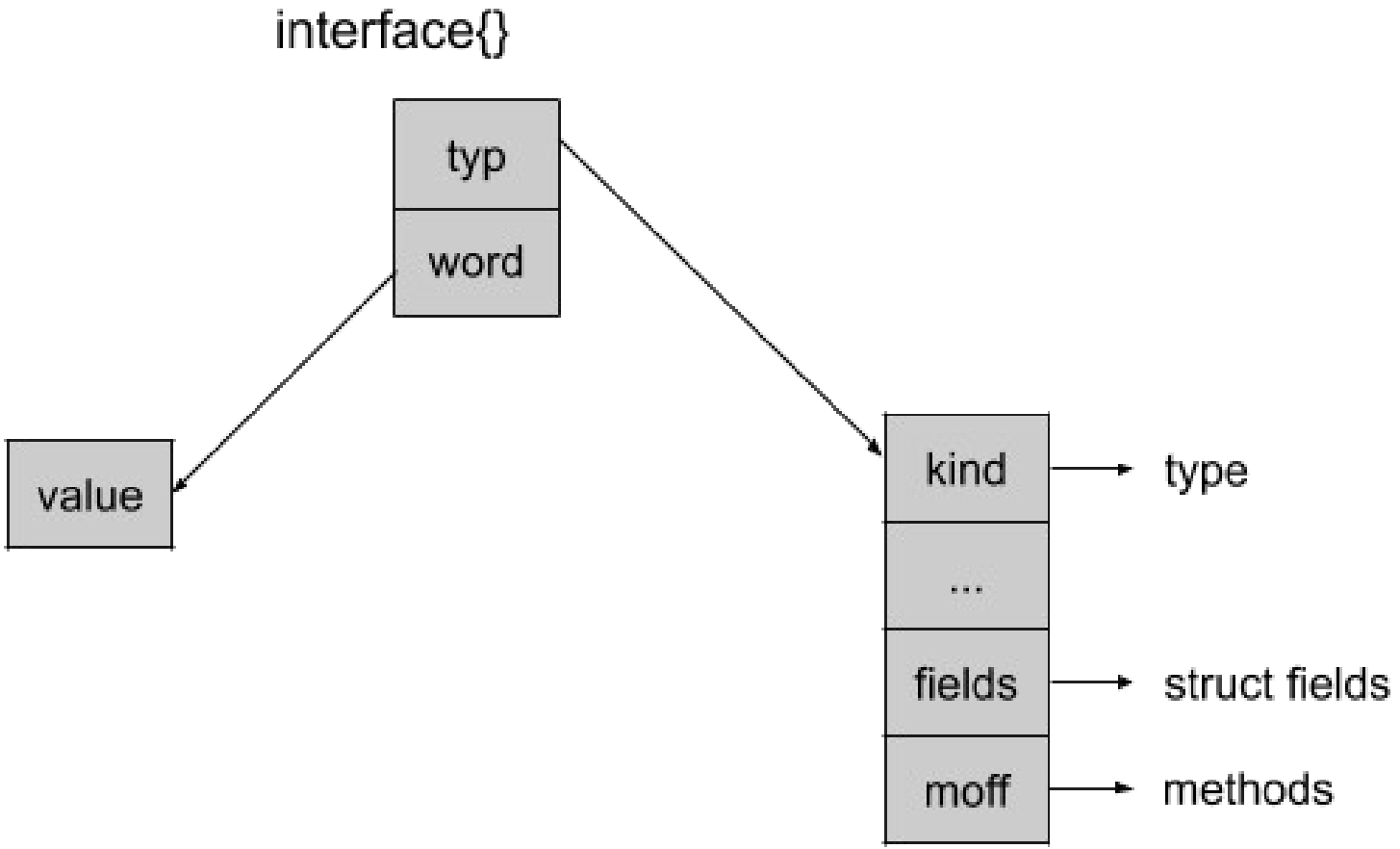
Рефлексия. Reflection

```
func main() {
   var i int8 = 1
   read(i)
}

func read(i interface{}) {
   println(i)
}

// (0x10591e0,0x10be5c6)
```

Рефлексия. Reflection

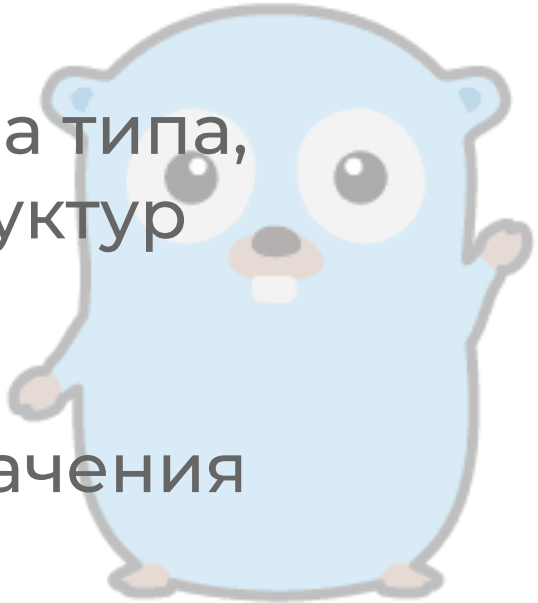


Рефлексия. Reflection

По сути в пакете "reflect" существует два вида рефлексии:

- рефлексия типа reflect.Type
  описывает исключительно свойства типа,
  а также список полей, методов структур

- рефлексия значения reflect.Value
  отражает свойства конкретного значения
  хранящегося в переменной



```
[
    {
      "title": "item 1",
      "price": 12
    },
    {
      "title": "item 1",
      "price": 8.93
    },
    {
      "title": "item 1",
      "price": "14"
    },
    {
      "title": "item 1",
      "price": "12.654"
    },
    {
      "title": "item 1",
      "price": 23
    }
]
```

```go
for _, item := range items {

	reflectType := reflect.TypeOf(item.Price)
	reflectValue := reflect.ValueOf(item.Price)

	switch reflectType.Kind() {

	case reflect.Int, reflect.Int8, reflect.Int16, reflect.Int32, reflect.Int64:
		totalPrice += float64(reflectValue.Int())

	case reflect.Uint, reflect.Uint8, reflect.Uint16, reflect.Uint32, reflect.Uint64:
		totalPrice += float64(reflectValue.Uint())

	case reflect.Float32, reflect.Float64:
		totalPrice += reflectValue.Float()

	case reflect.String:
		val, err := strconv.ParseFloat(reflectValue.String(), 64)
		if err != nil {
			panic(err.Error())
		}
		totalPrice += val

	default:
		panic(errors.New("Incorrect data provided!"))
	}
}

fmt.Println(totalPrice)
```