reflect2

used by 7.2k projects | go documentation | build failing | codecov 65% | go report A+

License Apache 2.0

reflect api that avoids runtime reflect.Value cost

- reflect get/set interface{}, with type checking
- reflect get/set unsafe.Pointer, without type checking
- `reflect2.TypeByName` works like `Class.forName` found in java

json-iterator use this package to save runtime dispatching cost. This package is designed for low level libraries to optimize reflection performance. General application should still use reflect standard library.

reflect2.TypeByName

```
// given package is github.com/your/awesome-package
type MyStruct struct {
    // ...
}

// will return the type
reflect2.TypeByName("awesome-package.MyStruct")
// however, if the type has not been used
// it will be eliminated by compiler, so we can not get it in runtime
```

reflect2 get/set interface{}

```
valType := reflect2.TypeOf(1)
i := 1
j := 10
valType.Set(&i, &j)
// i will be 10
```

to get set `type`, always use its pointer `*type`

reflect2 get/set unsafe.Pointer

```
valType := reflect2.TypeOf(1)
i := 1
j := 10
```

```
valType.UnsafeSet(unsafe.Pointer(&i), unsafe.Pointer(&j))
// i will be 10
```

to get set `type`, always use its pointer `*type`

benchmark

Benchmark is not necessary for this package. It does nothing actually. As it is just a thin wrapper to make go runtime public. Both `reflect2` and `reflect` call same function provided by `runtime` package exposed by go language.

unsafe safety

Instead of casting `[]byte` to `sliceHeader` in your application using unsafe. We can use reflect2 instead. This way, if `sliceHeader` changes in the future, only reflect2 need to be upgraded.

reflect2 tries its best to keep the implementation same as reflect (by testing).