-Converter

v2.6.2 ConverterConverter

:

```
func(T1) (T2, error)
```

T1 T2

(T1)

:

```
// RegisterConverter to register custom converter.
// It must be registered before you use this custom converting feature.
// It is suggested to do it in boot procedure of the process.
//
// Note:
//  1. The parameter `fn` must be defined as pattern `func(T1) (T2, error)
`.
//     It will convert type `T1` to type `T2`.
//  2. The `T1` should not be type of pointer, but the `T2` should be type
of pointer.
func RegisterConverter(fn interface{}) (err error)
```

```
package main

import (
        "fmt"

        "github.com/gogf/gf/v2/util/gconv"
)

type Src struct {
        A int
}

type Dst struct {
        B int
}

type SrcWrap struct {
        Value Src
}

type DstWrap struct {
        Value Dst
}

func SrcToDstConverter(src Src) (dst *Dst, err error) {
        return &Dst{B: src.A}, nil
}

// SrcToDstConverter is custom converting function for custom type.
func main() {
        // register custom converter function.
        err := gconv.RegisterConverter(SrcToDstConverter)
        if err != nil {
                panic(err)
        }

        // custom struct converting.
        var (
                src = Src{A: 1}
                dst *Dst
        )
        err = gconv.Scan(src, &dst)
        if err != nil {
                panic(err)
        }

        fmt.Println("src:", src)
        fmt.Println("dst:", dst)

        // custom struct attributes converting.
        var (
                srcWrap = SrcWrap{Src{A: 1}}
                dstWrap *DstWrap
        )
        err = gconv.Scan(srcWrap, &dstWrap)
        if err != nil {
                panic(err)
        }

        fmt.Println("srcWrap:", srcWrap)
        fmt.Println("dstWrap:", dstWrap)
}
```

:gconv.Scan

```
src: {1}
dst: &{1}
srcWrap: {{1}}
dstWrap: &{{1}}
```

gconv.Scangconv.ConvertWithRefer

```
package main

import (
        "fmt"

        "github.com/gogf/gf/v2/util/gconv"
)

type Src struct {
        A int
}

type Dst struct {
        B int
}

type SrcWrap struct {
        Value Src
}

type DstWrap struct {
        Value Dst
}

// SrcToDstConverter is custom converting function for custom type.
func SrcToDstConverter(src Src) (dst *Dst, err error) {
        return &Dst{B: src.A}, nil
}

func main() {
        // register custom converter function.
        err := gconv.RegisterConverter(SrcToDstConverter)
        if err != nil {
                panic(err)
        }

        // custom struct converting.
        var src = Src{A: 1}
        dst := gconv.ConvertWithRefer(src, Dst{})
        fmt.Println("src:", src)
        fmt.Println("dst:", dst)

        // custom struct attributes converting.
        var srcWrap = SrcWrap{Src{A: 1}}
        dstWrap := gconv.ConvertWithRefer(srcWrap, &DstWrap{})
        fmt.Println("srcWrap:", srcWrap)
        fmt.Println("dstWrap:", dstWrap)
}
```

Converterint, string

```
package main

import (
        "fmt"

        "github.com/gogf/gf/v2/os/gtime"
        "github.com/gogf/gf/v2/util/gconv"
)

type MyTime = *gtime.Time

type Src struct {
        A MyTime
}

type Dst struct {
        B string
}

type SrcWrap struct {
        Value Src
}

type DstWrap struct {
        Value Dst
}

// SrcToDstConverter is custom converting function for custom type.
func SrcToDstConverter(src Src) (dst *Dst, err error) {
        return &Dst{B: src.A.Format("Y-m-d")}, nil
}

// SrcToDstConverter is custom converting function for custom type.
func main() {
        // register custom converter function.
        err := gconv.RegisterConverter(SrcToDstConverter)
        if err != nil {
                panic(err)
        }

        // custom struct converting.
        var (
                src = Src{A: gtime.Now()}
                dst *Dst
        )
        err = gconv.Scan(src, &dst)
        if err != nil {
                panic(err)
        }

        fmt.Println("src:", src)
        fmt.Println("dst:", dst)

        // custom struct attributes converting.
        var (
                srcWrap = SrcWrap{Src{A: gtime.Now()}}
                dstWrap *DstWrap
        )
        err = gconv.Scan(srcWrap, &dstWrap)
        if err != nil {
                panic(err)
        }

        fmt.Println("srcWrap:", srcWrap)
        fmt.Println("dstWrap:", dstWrap)
}
```

```
type xxx = yyy*gtime.Time=int, string
```

```
src: {2024-01-22 21:45:28}
dst: &{2024-01-22}
srcWrap: {{2024-01-22 21:45:28}}
dstWrap: &{{2024-01-22}}
```

gconv.Scangconv.ConvertWithRefer

```
package main

import (
        "fmt"

        "github.com/gogf/gf/v2/os/gtime"
        "github.com/gogf/gf/v2/util/gconv"
)

type MyTime = *gtime.Time

type Src struct {
        A MyTime
}

type Dst struct {
        B string
}

type SrcWrap struct {
        Value Src
}

type DstWrap struct {
        Value Dst
}

// SrcToDstConverter is custom converting function for custom type.
func SrcToDstConverter(src Src) (dst *Dst, err error) {
        return &Dst{B: src.A.Format("Y-m-d")}, nil
}

// SrcToDstConverter is custom converting function for custom type.
func main() {
        // register custom converter function.
        err := gconv.RegisterConverter(SrcToDstConverter)
        if err != nil {
                panic(err)
        }

        // custom struct converting.
        var src = Src{A: gtime.Now()}
        dst := gconv.ConvertWithRefer(src, &Dst{})
        fmt.Println("src:", src)
        fmt.Println("dst:", dst)

        // custom struct attributes converting.
        var srcWrap = SrcWrap{Src{A: gtime.Now()}}
        dstWrap := gconv.ConvertWithRefer(srcWrap, &DstWrap{})
        fmt.Println("srcWrap:", srcWrap)
        fmt.Println("dstWrap:", dstWrap)
}
```