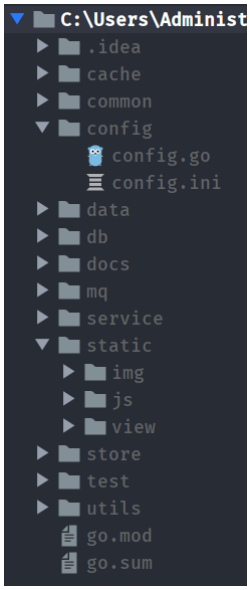
Go语言打包静态资源

1.我的项目结构



2.安装相关包

```
go get github.com/go-bindata/go-bindata/...
go get github.com/elazarl/go-bindata-assetfs/...
```

3.使用go-bindata-assetfs打包静态文件

使用下面命令即可在当前项目下生成 assets/asset.go，通过读取里面的内容就是读取静态资源了

```
mkdir assets -p && go-bindata-assetfs -pkg assets -o ./assets/asset.go
static/... config/config.ini
# -pkg xx 生成的文件，其package为xx
# -o xx   生成的文件名为xx
```

4.使用静态文件

配置文件的读取

- 原本读取配置文件

```
var Viper *viper.Viper // 供外部读取配置

func init() {
    Viper = viper.New()
    // 设置配置文件的名字
    Viper.SetConfigName("config")
    // 添加配置文件所在的路径
    Viper.AddConfigPath("./config")
    // 设置配置文件类型
    Viper.SetConfigType("ini")
    if err := Viper.ReadInConfig(); err != nil {
        panic(err)
    }
}
```

- 打包后这样读取配置

```
var Viper *viper.Viper // 供外部读取配置

func init() {
    Viper = viper.New()
    bytesData, err := assets.Asset("config/config.ini")
    if err != nil {
        panic(err)
    }
    Viper.SetConfigType("ini")
    if err = Viper.ReadConfig(bytes.NewBuffer(bytesData)); err != nil {
        panic(err)
    }
}
```

web静态资源的读取

- 比如gin原本这样提供静态文件

```
func Router() *gin.Engine {
    r := gin.Default()
    r.Static("/static/", "./static")
    return r
}
```

- 打包后

```
import (
    assetfs "github.com/elazarl/go-bindata-assetfs"
    "github.com/gin-gonic/contrib/static"
    "github.com/gin-gonic/gin"
    "net/http"
    "strings"
    "xcloud/assets"
)

type binaryFileSystem struct {
    fs http.FileSystem
}

func (b *binaryFileSystem) Open(name string) (http.File, error) {
    return b.fs.Open(name)
}

func (b *binaryFileSystem) Exists(prefix string, filepath string) bool {
    if p := strings.TrimPrefix(filepath, prefix); len(p) < len(filepath)
{
        if _, err := b.fs.Open(p); err != nil {
            return false
        }
        return true
    }
    return false
}
```

```
27
28 func BinaryFileSystem(root string) *binaryFileSystem {
29     fs := &assetfs.AssetFS{
30         Asset:     assets.Asset,
31         AssetDir:  assets.AssetDir,
32         AssetInfo: assets.AssetInfo,
33         Prefix:    root,
34     }
35     return &binaryFileSystem{fs: fs}
36 }
37
38 func Router() *gin.Engine {
39     r := gin.Default()
40     r.Use(static.Serve("/static/", BinaryFileSystem("static")))
41     return r
42 }
```