包和工具

导入包

导入路径

包——导入路径定位

```
import (
    // "标准库导入，直接包名，Go官方维护"
    "fmt"
    "math/rand"
    "encoding/json"

    // 第三方库导入，完整URL路径
    "golang.org/x/net/html"
    "github.com/go-sql-driver/mysql"

    // 本地项目导入
    // 在go.mod里，`module xxx.com/xxx`
    // "xxx.com/xxx/某个包"
)
```

本地包导入的相关介绍在ch2中，详见 谁教你的声明？.pdf

花活导入

别名

```
import (
    h "golang.org/x/net/html"
    sql "database/sql"
)

// import numpy as np? 😊
```

点导入

```
import . "math" // 可直接使用 Pow() 而非 math.Pow()
```

包声明

```
package main // package起手，main包我有
```

package 起手，后面跟 包名 ，默认标识符（我是哪个包的兵）

e.g. math/rand 包的每个源文件都是 package rand 起手，调用时就是 rand.xxx()

e.g. src/math/rand/rand.go

```go
// Copyright 2009 The Go Authors. All rights reserved.
// Use of this source code is governed by a BSD-style
// license that can be found in the LICENSE file.

// Package rand implements pseudo-random number generators suitable for tasks
// such as simulation, but it should not be used for security-sensitive work.
//
// Random numbers are generated by a [Source], usually wrapped in a [Rand].
// Both types should be used by a single goroutine at a time: sharing among
// multiple goroutines requires some kind of synchronization.
//
// Top-level functions, such as [Float64] and [Int],
// are safe for concurrent use by multiple goroutines.
//
// This package's outputs might be easily predictable regardless of how it's
// seeded. For random numbers suitable for security-sensitive work, see the
// crypto/rand package.
package rand

import (
        "internal/godebug"
        "sync"
        "sync/atomic"
        _ "unsafe" // for go:linkname
)

// A Source represents a source of uniformly-distributed
// pseudo-random int64 values in the range [0, 1<<63).
//
// A Source is not safe for concurrent use by multiple goroutines.
type Source interface {
        Int63() int64
        Seed(seed int64)
}
...
```

more source codes in https://github.com/golang/go/tree/master/src/math/rand

导入声明

多个导入

```
import "fmt"
import "os"
```

OK！√

```
import (
    "fmt"
    "os"
)
```

也OK！√(preferred)🥰

重名包

```
import (
    "crypto/rand"
    "math/rand"
)
```

都是 rand 怎么办？🤔

重名？重起个名！😋
还记得上面说的"别名"吗

```
import (
    "crypto/rand" // 你还叫rand
    mrand "math/rand" // 你叫mrand
    // 这下不就OK了？太会和稀泥了我的天😘
)
```

匿名导入

```
import _ "math/rand"
```

前面加了一个 _，表示匿名导入该包

匿名导入，不会直接使用其内容，而是

1. 执行包的 `init()` 函数（如果有）
2. 触发包的副作用（如驱动注册、全局变量初始化等）

纯纯工具包？😊

e.g.
`image` 包通过匿名导入扩展支持的格式

```
import _ "image/png"  // 添加 PNG 解码支持
import _ "image/jpeg" // 添加 JPEG 解码支持

func main() {
    f, _ := os.Open("photo.jpg")
    img, format, _ := image.Decode(f) // 自动识别格式
    fmt.Println("图像格式:", format)
}
```

注册数据库驱动

```
import _ "github.com/go-sql-driver/mysql"

func main() {
    db, err := sql.Open("mysql", "user:pass@/dbname") // 这里 "mysql" 已被驱动注册
    // ...
}
```

包的命名

敢教我起名字？你的...？

1. 简洁
2. 描述性
3. 无歧义
   temperatureconversion ？
   temp ??
   tempconv 😊

go tools?

```
go

# enter 'go' to query go tools
Go is a tool for managing Go source code.

Usage:

        go <command> [arguments]

The commands are:

        bug         start a bug report
        build       compile packages and dependencies
        clean       remove object files and cached files
        doc         show documentation for package or symbol
        env         print Go environment information
        fix         update packages to use new APIs
        fmt         gofmt (reformat) package sources
        generate    generate Go files by processing source
        get         add dependencies to current module and install them
        install     compile and install packages and dependencies
        list        list packages or modules
        mod         module maintenance
        work        workspace maintenance
        run         compile and run Go program
        telemetry   manage telemetry data and settings
        test        test packages
        tool        run specified go tool
        version     print Go version
        vet         report likely mistakes in packages

Use "go help <command>" for more information about a command.

Additional help topics:

        buildconstraint build constraints
        buildjson       build -json encoding
        buildmode       build modes
        c               calling between Go and C
        cache           build and test caching
```

```
        environment     environment variables
        filetype        file types
        goauth          GOAUTH environment variable
        go.mod          the go.mod file
        gopath          GOPATH environment variable
        goproxy         module proxy protocol
        importpath      import path syntax
        modules         modules, module versions, and more
        module-auth     module authentication using go.sum
        packages        package lists and patterns
        private         configuration for downloading non-public code
        testflag        testing flags
        testfunc        testing functions
        vcs             controlling version control with GOVCS

Use "go help <topic>" for more information about that topic.
```