

Packages

ภาษา Go นั้นถูกออกแบบมาเพื่อสนับสนุนแนวปฏิบัติที่ดี สำหรับการพัฒนาซอฟต์แวร์ โดยส่วนที่สำคัญของซอฟต์แวร์ที่มีคุณภาพสูงก็คือ การนำ code กลับมาใช้ซ้ำ (reuse) ซึ่งเป็นไปตามแนวคิด Don't Repeat Yourself (DRY)

ในบทที่ 7 เรื่อง Functions นั้นเป็นส่วนแรกของการ reuse เท่านั้น โดยภาษา Go ยังได้เตรียมกลไกหรือวิธีการอื่นๆ มาให้ด้วย คือ package ถ้าสังเกตจากโค้ดที่ผ่านมา จะพบว่า

```
import "fmt"
```

fmt คือชื่อ package ประกอบไปด้วยฟังก์ชันการทำงานที่เกี่ยวกับการจัดรูปแบบข้อมูล การแสดงผลลัพธ์กลับไปยังหน้าจอ การรวบรวมไว้ในที่เดียวกันนี้มีจุดประสงค์ 3 อย่าง ดังนี้

- ช่วยลดความซ้ำซ้อนของชื่อ ทำให้ชื่อของฟังก์ชันสั้นกระชับ
- ช่วยให้ง่ายต่อการค้นหาโค้ดที่ต้องการใช้ซ้ำ
- ช่วยทำให้ compiler ทำงานที่รวดเร็วขึ้น เนื่องจากจะ compile ในเฉพาะส่วน หรือใน package เท่านั้น

การสร้าง Packages

Package นั้นเป็นแนวคิดพื้นฐานสำหรับการแบ่งกลุ่มของโปรแกรม ดังนั้นเรามาดูกันว่าสามารถสร้างมันได้อย่างไร เริ่มต้นสร้างโฟลเดอร์ ชื่อ chapter11 ใน ~/Go/src/golang-book แล้วสร้างไฟล์ชื่อ main.go ซึ่งมีโค้ดดังนี้

```
package main

import "fmt"
import "golang-book/chapter11/math"

func main() {
    xs := []float64{1,2,3,4}
    avg := math.Average(xs)
    fmt.Println(avg)
}
```

ต่อมาทำการสร้างโฟลเดอร์ ชื่อ math ใน chapter11 โดยภายในโฟลเดอร์ math สร้างไฟล์ชื่อ math.go มีโค้ดดังนี้

```
package math
```

```
func Average(xs []float64) float64 {
    total := float64(0)
    for _, x := range xs {
        total += x
    }
    return total / float64(len(xs))
}
```

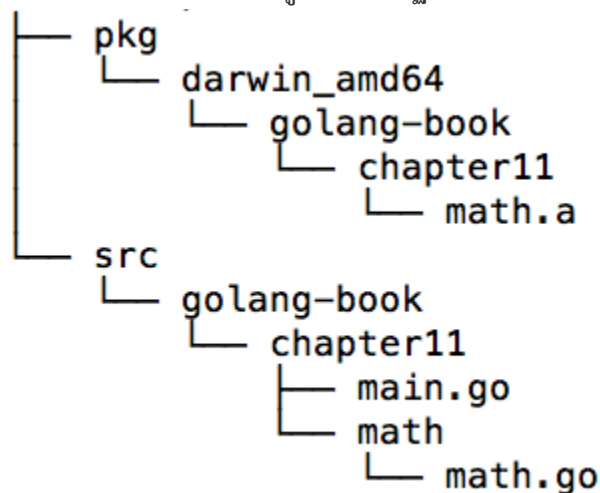
ไปยัง terminal หรือ command line แล้วเข้าไปที่โฟลเดอร์ math พิมพ์คำสั่ง go install

```
$cd math
```

```
$go install
```

Go จะทำการ compile ไฟล์ math.go และสร้างไฟล์ linkable object ดังนี้ `~/Go/pkg/os_arch/golang-book/chapter11/math.a`

โดยชื่อ `<os_arch>` ขึ้นอยู่กับระบบปฏิบัติการที่ใช้งาน ตัวอย่างเช่น `darwin_amd64` ดังรูป



หมายเหตุ

ถ้า run แล้วมีปัญหาหรือแสดงข้อความลักษณะนี้

go install: no install location for directory ... outside GOPATH

ให้ทำการแก้ไขด้วยการกำหนดตัวแปรระบบชื่อว่า `GOPATH` ไปดังนี้

```
$export GOPATH=~/Go/
```

ต่อมาให้กลับไปยังโฟลเดอร์ chapter11 แล้วพิมพ์คำสั่ง go run main.go ดังนี้

```
$cd ..
```

```
$go run main.go
```

จะแสดงค่า 2.5 ขึ้นมา
เรามาดูส่วนที่น่าสนใจกัน

- `math` คือชื่อ `package` ซึ่งถ้าไปดูในเอกสารจะพบว่ามันคือชื่อ `library` พื้นฐานของ Go แต่โครงสร้างของ `package` ทำให้เราสามารถใช้งาน `math` ได้ เนื่องจาก `package math` ของ Go นั้นคือ `math` แต่ `math` ของเราหรือจากตัวอย่างอยู่ที่ `golang-book/chapter11/math` ซึ่งจะเห็นจากภาพด้านบน
- เมื่อต้องการ `import` ใช้งาน `math` ของเราต้องใช้ชื่อเต็มดังนี้ `import "golang-book/chapter11/math"`
- เราสามารถใช้งานผ่านชื่อสั้นๆ เพื่อเรียกใช้งานฟังก์ชันจาก `library` ต่างๆ ได้ ด้วย Go อนุญาตให้ใช้การ `alias` ดังโค้ดด้านล่าง

```
import m "golang-book/chapter11/math"
```

```
func main() {  
    xs := []float64{1,2,3,4}  
    avg := m.Average(xs)  
    fmt.Println(avg)  
}
```

`m` คือชื่อที่ต้องการใช้งาน

- คุณอาจตั้งข้อสังเกตว่า ในทุกๆ ฟังก์ชันใน `package` จะขึ้นต้นด้วยตัวอักษรพิมพ์ใหญ่เสมอ ซึ่งตรงนั้นมันมีความหมาย กล่าวคือ เมื่อใดก็ตามที่ชื่อฟังก์ชันขึ้นต้นด้วยตัวอักษรพิมพ์ใหญ่แล้ว `package` อื่นๆ สามารถเห็นและใช้งานฟังก์ชันนั้นได้
- การจัดการเรื่องการใช้งานของ `package` นั้นถือว่าเป็นแนวปฏิบัติที่ดี ที่จะให้ `package` อื่นใช้งานได้หรือซ่อนก็ได้ ทำให้เราสะดวกต่อการใช้งานและการแก้ไขส่วนต่างๆ ใน `package` โดยไม่กระทบต่อส่วนอื่นๆ ด้วย
- เพื่อให้ง่ายต่อชีวิต แนะนำให้ชื่อ `package` นั้นควรตรงกับชื่อไฟล์เดอร์

เอกสาร

Go นั้นมีความสามารถในการสร้างเอกสารสำหรับ package แบบอัตโนมัติอยู่แล้ว เพียงแค่ใช้คำสั่ง

```
$godoc golang-book/chapter11/math Average
```

ผลการทำงานเป็นดังรูป

PACKAGE DOCUMENTATION

```
package math
import "golang-book/chapter11/math"
```

FUNCTIONS

```
func Average(xs []float64) float64
```

ซึ่งดูแล้วไม่สวย เนื่องจากเรายังไม่ใส่ข้อความ comment อะไรลงไป ใน code ดังนั้นลองเพิ่ม comment ไปใน code หน่อยสิ ดังนี้

```
// Finds the average of a series of numbers
func Average(xs []float64) float64 {
```

แล้วลอง run godoc ใหม่จะได้ผลดังรูป ซึ่งดูดีขึ้นมาน้อย

PACKAGE DOCUMENTATION

```
package math
import "golang-book/chapter11/math"
```

FUNCTIONS

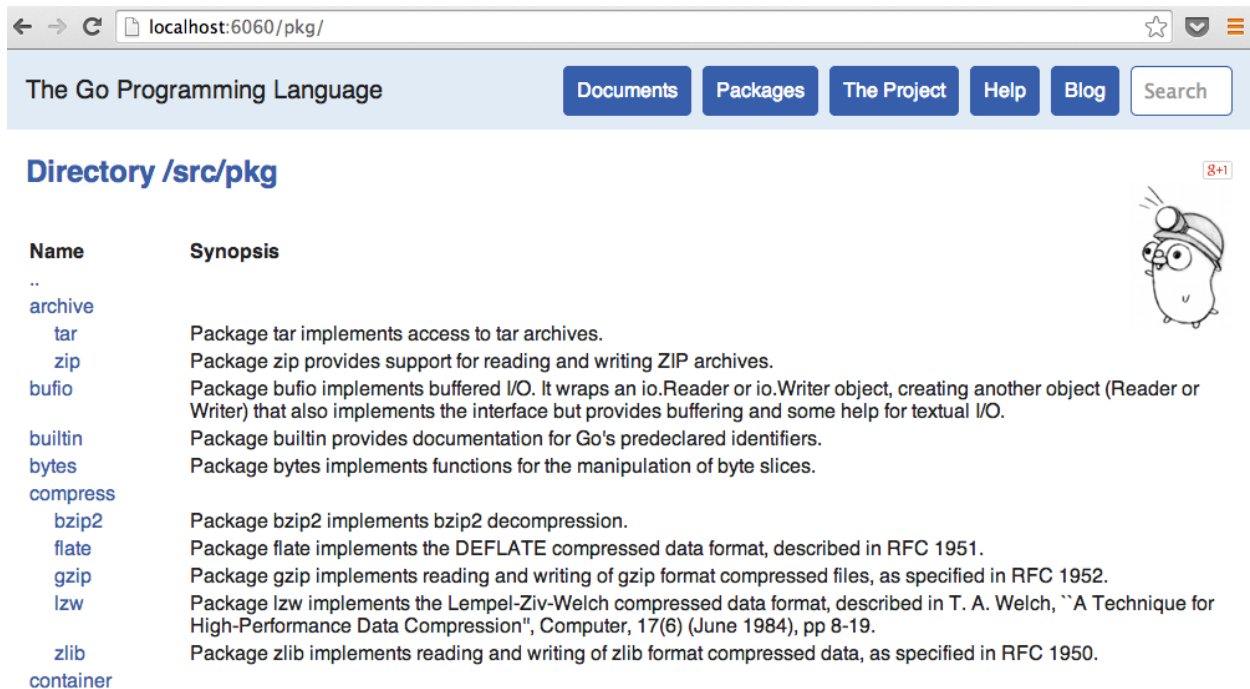
```
func Average(xs []float64) float64
    Finds the average of a series of numbers
```

เมื่อทำการแก้ไขไฟล์ math.go แล้วอย่าลืมพิมพ์คำสั่ง go install ก่อนที่จะสั่ง godoc นะ
ต่อไปทำการแสดงเอกสารผ่านเว็บดีกว่า เพราะว่ามันจะมีประโยชน์มากขึ้น โดยใช้คำสั่งดังนี้
\$godoc -http=":6060"

แล้วเข้า browser ไปยัง url นี้

<http://localhost:6060/pkg/>

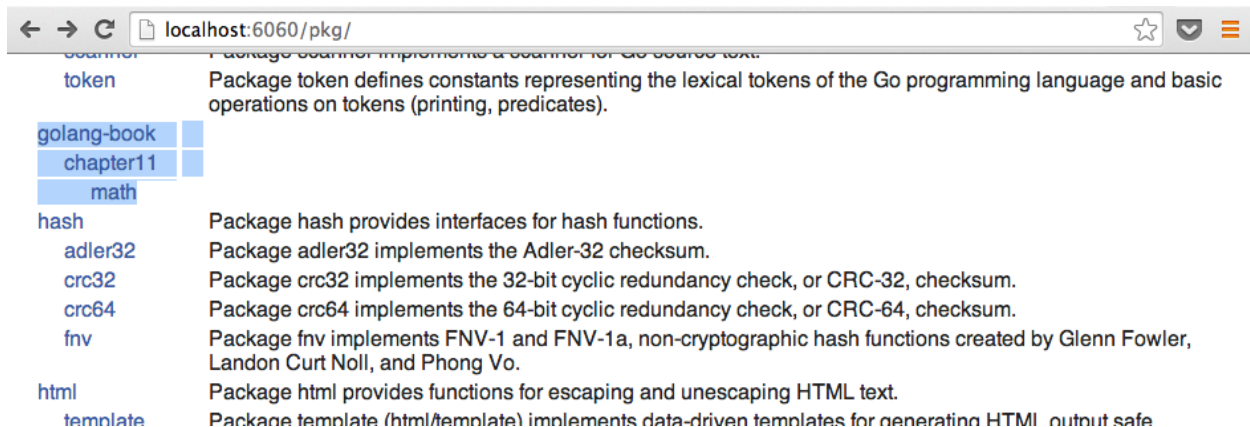
ผลการทำงานเป็นดังนี้



The screenshot shows the Go Programming Language website at localhost:6060/pkg/. The page has a navigation bar with links: Documents, Packages, The Project, Help, Blog, and a Search button. The main heading is "Directory /src/pkg". On the right, there is a cartoon character wearing a hard hat and a small "8+1" badge. The main content is a table with two columns: "Name" and "Synopsis".

Name	Synopsis
..	
archive	
tar	Package tar implements access to tar archives.
zip	Package zip provides support for reading and writing ZIP archives.
bufio	Package bufio implements buffered I/O. It wraps an io.Reader or io.Writer object, creating another object (Reader or Writer) that also implements the interface but provides buffering and some help for textual I/O.
builtin	Package builtin provides documentation for Go's predeclared identifiers.
bytes	Package bytes implements functions for the manipulation of byte slices.
compress	
bzip2	Package bzip2 implements bzip2 decompression.
flate	Package flate implements the DEFLATE compressed data format, described in RFC 1951.
gzip	Package gzip implements reading and writing of gzip format compressed files, as specified in RFC 1952.
lzv	Package lzv implements the Lempel-Ziv-Welch compressed data format, described in T. A. Welch, "A Technique for High-Performance Data Compression", Computer, 17(6) (June 1984), pp 8-19.
zlib	Package zlib implements reading and writing of zlib format compressed data, as specified in RFC 1950.
container	

เมื่อเลื่อนลงมาจะเห็น package /golang-book/chapter11/math ดังรูป



The screenshot shows the Go Programming Language website at localhost:6060/pkg/. The page has a navigation bar with links: Documents, Packages, The Project, Help, Blog, and a Search button. The main heading is "Directory /src/pkg". On the right, there is a cartoon character wearing a hard hat and a small "8+1" badge. The main content is a table with two columns: "Name" and "Synopsis".

Name	Synopsis
scanner	Package scanner implements a scanner for Go source text.
token	Package token defines constants representing the lexical tokens of the Go programming language and basic operations on tokens (printing, predicates).
golang-book	
chapter11	
math	
hash	Package hash provides interfaces for hash functions.
adler32	Package adler32 implements the Adler-32 checksum.
crc32	Package crc32 implements the 32-bit cyclic redundancy check, or CRC-32, checksum.
crc64	Package crc64 implements the 64-bit cyclic redundancy check, or CRC-64, checksum.
fnv	Package fnv implements FNV-1 and FNV-1a, non-cryptographic hash functions created by Glenn Fowler, Landon Curt Noll, and Phong Vo.
html	Package html provides functions for escaping and unescaping HTML text.
template	Package template (html/template) implements data-driven templates for generating HTML output safe

เข้าไปดูใน package math มีข้อมูลดังรูป

Package math

8+1

```
import "golang-book/chapter11/math"
```

Overview
Index

Overview ▾

Index ▾

func Average(xs []float64) float64

Click to hide Index section

Package files

math.go

func Average

```
func Average(xs []float64) float64
```

Finds the average of a series of numbers

Build version go1.2.1.

Except as noted, the content of this page is licensed under the Creative Commons Attribution 3.0 License, and code is licensed under a [BSD license](#).
[Terms of Service](#) | [Privacy Policy](#)

ปัญหา

- ทำไมเราต้องใช้ package ละ
- ความแตกต่างระหว่าง Average กับ average คืออะไร
- package alias คืออะไร และใช้มันอย่างไร
- ให้ทำการคัดลอกฟังก์ชัน average จากบทที่ 7 และทำการสร้างฟังก์ชัน Min และ Max ซึ่งใช้สำหรับหาค่าตัวเลขน้อยที่สุดและมากที่สุดจาก Slice มีชนิดเป็น float64
- สร้างเอกสารของ 3 ฟังก์ชันอย่างไร
-