디스커버리 Go 2 - 환경설정 및 시작하기

Posted on May 7, 2016 by in

이 포스트는 `디스커버리 GO 언어` 2장을 읽고 정리한 것이며 작업 환경은 `Ubuntu 16.04` 입니다.

Go 설치하기

Window나 Mac의 경우는 golang.org에서 설치 파일을 다운로드 받아 쉽게 설치가 가능합니다.

Ubuntu의 경우에는 `apt-get` 으로 설치합니다.

```
$ sudo apt-get install golang
$ go version
```

GOPATH 설정

`GOPATH` 란 작업 파일들이 들어갈 디렉토리를 의미하며 사용자의 프로젝트 뿐만 아니라 수많은 오픈 소스 코드들을 내려 받을 경우에 저장되는 공간이기도 합니다.

아래 예제에서는 `/home/$USERNAME/workspaces/go` 을 GOPATH로 설정합니다.

```
$ mkdir -p ~/workspace/go
# 아래 환경변수 선언은 .bashrc에 작성하는 것이 좋음
$ export GOPATH=/home/$USERNAME/workspaces/go
$ export GOBIN=$GOPATH/bin
$ export PATH=$PATH:$GOBIN
```

GOPATH내에는 아래 3개의 디렉토리가 생성되며 각각의 역할은 아래와 같습니다.

- **bin**: 실행파일들이 들어감
- **pkg**: Package Object파일들이 들어감(라이브러리)
- **src**: 소스 코드들이 들어간다

오픈소스를 다운로드 할 경우에 `pkg` 와 `src` 의 하위 디렉토리는 주소와 비슷하게 구성됩니다.

labix.org/v2/mgo 에 있는 몽고디비 접근용 라이브러리를 가져왔을 경우에 디렉토리 구조는 아래와 같아집니다.

- src/labix.org/v2/mgo
- pkg/labix.org/v2/mgo

HelloWorld

Go로 첫 소스를 만들어보자.

```
# package 디렉토리 생성(amazingguni는 내 계정이다.)
$ mkdir -p $GOPATH/src/github.com/amazingguni/gogo/hello
$ cd $GOPATH/src/github.com/amazingguni/gogo/hello
$ vim hello.go
```

```
// hello.go
package main

import "fmt"

func main() {
    fmt.Println("Hello, playground")
}
```

`go run` 으로 작성한 소스 코드를 실행할 수 있다.

```
$ go run hello.go
Hello, playground
```

`go install` 명령어로 프로그램을 설치할 수 있으며 binary파일은 *$GOPATH/bin* 에 저장된다.

```
$ echo $PWD
$GOPATH/src/github.com/amazingguni/gogo/hello
$ go install ./
$ hello
Hello, playground
```

go get

`go get` 으로 자신이 작성하지 않고도 다른 사람이 공유하는 소스 코드를 받을 수 있습니다.

```
$ go get github.com/jaeyeom/gogo/hanoi
$ go install github.com/jaeyeom/gogo/hanoi
$ hanoi
Number of disks: 3
1 -> 2
1 -> 3
2 -> 3
1 -> 2
3 -> 1
3 -> 2
1 -> 2
```

자주 쓰는 명령 도구 설치

Go 언어를 사용할 때 자주 쓰게 자주 쓰는 tool들은 아래 명령어로 간편하게 설치할 수 있습니다.

```
$ go get golang.org/x/tools/cmd/...
```

각각의 도구는 아래와 같은 역할을 합니다.

- **goimports**: 자동으로 import 경로를 추론하여 소스 코드의 상단에 추가해주는 도구
- **gofmt**: 소스 코드의 Format을 맞춰주는 도구
- **godoc**: Go 프로그램의 문서를 볼 수 있는 도구
- **Oracle**: 소스 코드에 대해 여러가지를 알려주는 강력한 도구(보통 다른 툴과 연계해서 사용)

- **Vet**: 소스 코드 검사 도구
- **Fix**: deprecated된 API호출 등을 자동으로 고쳐주는 도구
- **Test**: 테스트를 수행하는 도구

gofmt

`gofmt` 는 format을 맞춰주는 도구입니다.

formatting이 되어있지 않는 아래 코드를 작성한 이후에

```
package main

import "fmt"

func main() {
fmt.Println("Hello, playground")
}
```

`gofmt` 를 사용하면 format에 맞게 수정해줍니다.

```
$ gofmt -w hello.go
```

```
package main

import "fmt"

func main() {
    fmt.Println("Hello, playground")
}
```

goimports

`goimports` 는 import 경로를 추론하여 소스 상단에 추가를 해주는 도구입니다.

Eclipse에서 `ctrl+shift+o` 로 사용했던 optimized import 기능과 동일합니다.

아래 예제에서는 위와 달리 import "fmt"가 빠져있습니다.

```
package main

func main() {
    fmt.Println("Hello, playground")
}
```

그대로 실행하면 에러가 발생합니다.

```
$ go run hello.go
# command-line-arguments
./hello.go:4: undefined: fmt in fmt.Println
```

아래와 같이 goimports를 사용하면 fmt package가 추가되는 것을 알 수 있습니다.

```
$ goimports -w hello.go
```