DIGITALHOUSE

Organización de nuestro código

Primer acercamiento a Go

Certified Tech Developer

The Ultimate Degree

Índice

01 Packages

Packages

Recordemos: cada archivo de Go pertenece a un package. Entonces, para declarar uno de estos archivos usamos la siguiente sintaxis:

```
{}  package packagename
```

- La carpeta que contiene los distintos archivos del package y el package se deben llamar igual a este último.
- La declaración del package anterior debe ser la primera línea de código en el archivo fuente de Go.
- Todas las funciones, tipos y variables definidas en el archivo fuente de Go pasan a formar parte del paquete declarado.
- El nombre de los packages debería ser una sola palabra y todo en minúscula.

Package y función main

Los programas en Go comienzan a ejecutarse en el **package main**. Este es un package que se usa con programas que están destinados a ser ejecutables. Se los conoce como comandos. Los otros programas se denominan simplemente packages.

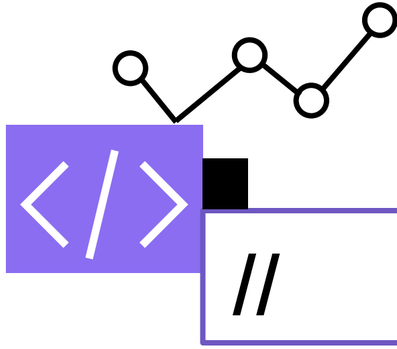
```
{}
```

```
package main
```

La función **main()** es especial, ya que es el punto de entrada de un programa ejecutable. Veamos un ejemplo de un programa ejecutable en Go.

```
{}
```

```
func main() {
  // Tu código
}
```



Importaciones de packages

Go nos proporciona dos formas de importar packages. Veamos cuáles son:

- Importación de packages individual:

```
{}
import "fmt"
import "time"
```

- Importación de packages grupal

```
{}
import (
    "fmt"
    "time"
)
```

02 Modules

¿Cómo inicializar un module?

Antes de iniciar un módulo Go, tenemos que asegurarnos de crear un nuevo repositorio en GitHub —u otros controles de versión (por ejemplo, GitLab)—. Luego, podremos clonar el repositorio. Para iniciar un módulo Go usamos este comando:

Después de que se haya iniciado el módulo, se crea el archivo **.go.mod**. Este contiene las dependencias que se utilizan en la aplicación.

¿Cómo agregar una dependencia?

Se puede agregar una dependencia a la aplicación usando el comando **go get**. Go permite agregarle el flag **-u** para descargarla si no existe o actualizar la dependencia.

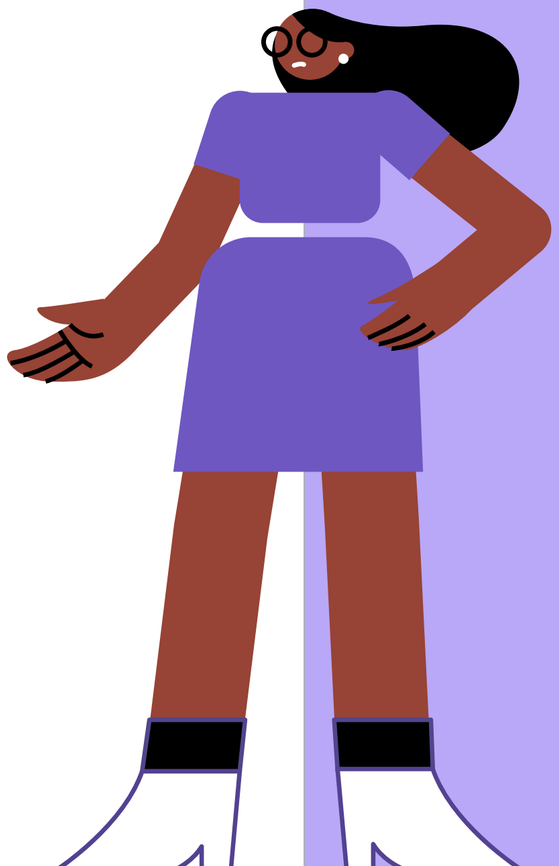
```
{}
go get -u github.com/gin-gonic/gin
```

Luego de agregar la dependencia, se puede utilizar dentro de nuestros archivos del módulo agregando el **import**.

```
{}
import (
    "fmt"
    "github.com/gin-gonic/gin"
)
```

Conclusiones

Aprendimos cómo crear packages e importarlos, y vimos la importancia de la función main en un proyecto. Además, nos focalizamos en la inicialización de **modules** para el buen manejo de dependencias. Por último, pudimos usar el comando **go get** para agregar dependencias a un proyecto. Todo esto nos ayudará a tener una correcta organización del código.



DIGITALHOUSE

¡Muchas gracias!