Using Other Packages

Michael Van Sickle

@vansimke

Overview

Typical imports

Alternative import methods

The vendor directory

Typical imports

```
package main

import “fmt”

func main() {
        fmt.Println(“Hello,gophers!”)
}
```

Typical imports

```
package main

import (
        “log”
        “net/http”
)

func main() {
        err := http.ListenAndServe(“:8080”, nil)
        if err != nil {
                log.Fatal(err)
        }
}
```

Alternative Import Methods

Aliases

Import for side effect

Internal packages

Relative imports

Aliasing

```
package main

import (
        “encoding/json”
)

func main() {
        ...
        data, err := json.Marshal(...)
        ...
}
```

Aliasing

```
package main

import (
        “encoding/json”
        “pluralsight.com/libmanager/json”
)

func main() {
        ...
        data, err := json.Marshal(...)
        ...
}
```

Aliasing

```
package main

import (
        “encoding/json”
        lmjson “pluralsight.com/libmanager/json”
)

func main() {
        ...
        data, err := json.Marshal(...)
        ...
}
```

Import for Side Effect

```
import (
        “database/sql”
    _ “github.com/lib/pq”
)
```

calls init() functions

Internal Packages

Provides another scoping mechanism

Accessible by parent and its children

Enables better organization without leaking details

Relative Imports

Import package relative to current one

Not valid in workspaces or modules

Probably will never need this!

Vendor Directories

Workspaces only

First official version management strategy

Hierarchically resolved

Summary

Typical imports

Alternative imports

- aliases
- import for side effect
- internal packages
- relative imports

The vendor directory

- workspaces only!