Golang Training Course

Ngày: 1st June 2021

Outline

Input

- Golang Fundamentals:
  - Arrays, Slices
  - Maps
  - Structs
  - If/Else/Switch
  - For/Loop
  - Function
  - Sort
  - Defer
  - Pointer
  - Type
  - Iota
  - Interface
  - Error
  - Modules/Package/Imports
- Sample Application

Output

- Viết được một chương trình sử dụng Maps làm DataStore. Thực hiện các thao tác CRUD (Create, Update, Read, Delete)

Content

People Store Application

Chương trình này sẽ Cover toàn bộ nội dung bài học Chapter 2

Bắt đầu:

- Khởi tạo dự án tên là `go-training` (Có thể dùng `mkdir $GOPATH/src/go-training`)
- Thực hiện chạy `go mod init`

Thêm Code:

- `entity/people.go`:

```go
package entity

import "fmt"

type People struct {
  Id      string
  Name    string
  Age     int
  Company string
  Address string
}

func (p People) SayHello() {
  fmt.Printf("Hello, I'm %s, %d years old\n", p.Name, p.Age)
}

func (p *People) UpdateAddress(address string) {
  p.Address = address
}

func (p *People) UpdateCompany(company string) {
  p.Company = company
}

func (p People) ToString() string {
  return fmt.Sprintf("Id: %s, Name: %s, Age: %d, Company: %s, Address: 
%s", p.Id, p.Name, p.Age, p.Company, p.Address)
}
```

- `internal/const.go`:

```go
package internal

type Order int

const (
  Asc Order = iota
  Desc
)
```

- `internal/int.go`:

```go
package internal

import (
  "math/rand"
)

func RandomInt(min, max int) int {
  return rand.Intn((max - min) + min)
}
```

- `internal/string.go`

```go
package internal

import "github.com/google/uuid"

func UUID() string {
  return uuid.New().String()
}
```

- `db/datastore.go`

```go
package db

import (
  "errors"
  "fmt"
  "go-training/entity"
  "go-training/internal"
  "sort"
)

type DataStore struct {
  store  map[string]*entity.People
  length int
}

func NewDataStore(rows int) (*DataStore, error) {
  if rows <= 0 {
    return nil, errors.New("rows must be > 0")
  }
  return &DataStore{
    store:  make(map[string]*entity.People, rows),
    length: rows,
  }, nil
}

func (ds *DataStore) Add(key string, val *entity.People) error {
  if key == "" {
    return errors.New("key must not be empty")
  }
  if _, ok := ds.store[key]; ok {
    return fmt.Errorf("key: %s is duplicated, the value need a unquie
key", key)
  }
  if len(ds.store) == ds.length {
    return fmt.Errorf("the data store is full, only store %d rows", 
ds.length)
  }

  ds.store[key] = val
  return nil
}

func (ds *DataStore) Update(key string, val *entity.People) error {
  if _, ok := ds.store[key]; !ok {
    return fmt.Errorf("key: %s is not existed to update", key)
  }
  ds.store[key] = val
  return nil
}

func (ds *DataStore) Read(key string) (*entity.People, error) {
  if _, ok := ds.store[key]; !ok {
    return nil, fmt.Errorf("key: %s is not existed to read", key)
  }
  return ds.store[key], nil
}

func (ds *DataStore) Delete(key string) error {
  if _, ok := ds.store[key]; !ok {
    return fmt.Errorf("key: %s is not existed to delete", key)
  }
  delete(ds.store, key)
  return nil
}

func (ds *DataStore) PrintAsOrder(order internal.Order) {
  keys := make([]string, 0)
  for k := range ds.store {
    keys = append(keys, k)
  }
  switch order {
  case internal.Asc:
    sort.Strings(keys)
  case internal.Desc:
    sort.Sort(sort.Reverse(sort.StringSlice(keys)))
  }

  for _, v := range keys {
    fmt.Println(ds.store[v].ToString())
  }
}
```

- `cmd/store.go`

```
package cmd
```

```go
import (
  "errors"
  "go-training/db"
  "go-training/entity"
  "go-training/internal"
)

type Store struct {
  bootstraped bool
  DataStore   *db.DataStore
}

func NewStore(rows int) (*Store, error) {
  ds, err := db.NewDataStore(rows)
  if err != nil {
    return nil, err
  }
  return &Store{
    DataStore: ds,
  }, nil
}

func (s *Store) Free() {
  s.DataStore = nil
}

func (s *Store) Bootstrap(rows int) error {
  if s.bootstraped {
    return errors.New("data is bootstraped")
  }
  for i := 0; i < rows; i++ {
    key := internal.UUID()
    // No need to handle error because this is bootstrap
    s.DataStore.Add(key, &entity.People{
      Id:      key,
      Name:    key,
      Age:     internal.RandomInt(23, 50),
      Company: "FPT Software",
      Address: "17 Duy Tan",
    })
  }
  return nil
}

func (s *Store) AddPeople(p *entity.People) error {
  if p == nil {
    return errors.New("people must not be empty")
  }
  if p.Name == "" {
    return errors.New("name must not be empty")
  }
  if p.Age < 23 {
    return errors.New("age must > 23")
  }
  return s.DataStore.Add(p.Id, p)
}

func (s *Store) SearchPeople(id string) (*entity.People, error) {
  return s.DataStore.Read(id)
}

func (s *Store) DeleteById(id string) error {
  return s.DataStore.Delete(id)
}

func (s *Store) UpdateById(id string, p *entity.People) error {
  return s.DataStore.Update(id, p)
}

func (s *Store) PrintAsOrder(order internal.Order) {
  s.DataStore.PrintAsOrder(order)
}
```

- `main.go`

```go
package main

import (
  "fmt"
  "go-training/cmd"
  "go-training/entity"
  "go-training/internal"
  "log"
  "math/rand"
  "time"
)

func init() {
  rand.Seed(time.Now().UnixNano())
}

func main() {
  s, err := cmd.NewStore(100)
  if err != nil {
    log.Fatal(err.Error())
  }
  defer s.Free()

  _ = s.Bootstrap(10)
  id_1 := internal.UUID()

  if err := s.AddPeople(&entity.People{
    Id:      id_1,
    Name:    "Nguyen Van A",
    Age:     30,
    Company: "FPT Software",
    Address: "17 Duy Tan",
  }); err != nil {
    log.Fatal(err.Error())
  }

  fmt.Println("Print Asc Order")
  s.PrintAsOrder(internal.Asc)
}
```

- Khởi chạy: `go mod tidy`
- Kiểm tra `go.mod` (Phải có thêm thư viện UUID) và đã tạo ra `go.sum`

```
module go-training

go 1.16

require github.com/google/uuid v1.2.0
```

- Khởi chạy với `go run main.go`
- Kết quả Output có thể:

```
Print Asc Order
Id: 0665ccf7-262f-4e91-a944-1c25f9086214, Name: 0665ccf7-262f-4e91-a944-1c25f9086214, Age: 28, Company: FPT Software, Address: 17 Duy Tan
Id: 1004f6f5-6c15-435f-afce-3c1c78bf2a5d, Name: 1004f6f5-6c15-435f-afce-3c1c78bf2a5d, Age: 39, Company: FPT Software, Address: 17 Duy Tan
Id: 3519034e-d79d-4c1b-8971-9512287debd1, Name: Nguyen Van A, Age: 30, Company: FPT Software, Address: 17 Duy Tan
Id: 55697257-e700-464e-a918-5717118f7ddb, Name: 55697257-e700-464e-a918-5717118f7ddb, Age: 46, Company: FPT Software, Address: 17 Duy Tan
Id: 5b161891-eca6-4950-8a3a-1aa600b407df, Name: 5b161891-eca6-4950-8a3a-1aa600b407df, Age: 2, Company: FPT Software, Address: 17 Duy Tan
Id: 7c2b2192-dea5-41dd-97e2-851717b286e4, Name: 7c2b2192-dea5-41dd-97e2-851717b286e4, Age: 7, Company: FPT Software, Address: 17 Duy Tan
Id: 9420cc02-9eac-4d0c-98d1-f2d1e969758a, Name: 9420cc02-9eac-4d0c-98d1-f2d1e969758a, Age: 11, Company: FPT Software, Address: 17 Duy Tan
Id: ab87df60-22ea-42bf-9c5c-13b0fa1fc7a3, Name: ab87df60-22ea-42bf-9c5c-13b0fa1fc7a3, Age: 14, Company: FPT Software, Address: 17 Duy Tan
Id: c2b4f25b-1d33-482b-9066-86a2f0dca5e7, Name: c2b4f25b-1d33-482b-9066-86a2f0dca5e7, Age: 15, Company: FPT Software, Address: 17 Duy Tan
Id: e4b7fd54-0d17-4366-b756-4cb3d733f8cb, Name: e4b7fd54-0d17-4366-b756-4cb3d733f8cb, Age: 26, Company: FPT Software, Address: 17 Duy Tan
Id: f7e9e4fd-fc42-4663-8dc2-98bf86f8575a, Name: f7e9e4fd-fc42-4663-8dc2-98bf86f8575a, Age: 40, Company: FPT Software, Address: 17 Duy Tan
```

TODO

- Thực hiện viết 1 chương trình tương tự, hoặc sử dụng các hàm làm quen như Delete, Update, Read,...