TradeMarkia_Challenge

Name : Mahendra Kumar

Registration Number : 20BCY10019

Part 1 (Data)

- Downloaded the ‘tt230101.xml’ file from the provided link.

reading from the XML file:

```
file, err := os.Open("tt230101.xml")
  if err != nil {
    log.Fatalf(err.Error())
  }
  defer file.Close()

  // Read the XML data
  data, err := io.ReadAll(file)
  if err != nil {
    log.Fatalf(err.Error())
  }
```

- Unmarshal the data into bytes by creating a schema that reads XML.

first creating a go program xml_file.go to make structure of the given XML data to umarshal it using ‘encoding/xml’ package and marshal it into json format using ‘encoding/json’.

```
package datatype

type XmlFile struct {
  Version struct {
    Version_no  string `xml:"version-no"  json:"version-no"`
    VersionDate uint   `xml:"version-date"  json:"version-date"`
  } `xml:"version" json:"version"`
  Action_key_code       string                `xml:"action-key-code" json:"action-key-code"`
  TransactionDate       uint                  `xml:"transaction-date" json:"transaction-date"`
  Proceedinginformation Proceedinginformation `xml:"proceeding-information"  json:"proceeding-information"`
}

type Proceedinginformation struct {
  ProceedingEntry []ProceedingEntry `xml:"proceeding-entry"  json:"proceeding-entry"`
}

type ProceedingEntry struct {
  Number                    uint   `xml:"number" json:"number"`
  Typecode                  string `xml:"type-code" json:"type-code"`
  FilingDate                uint   `xml:"filing-date" json:"filing-date"`
  EmployeeNumber            uint   `xml:"employee-number" json:"employee-number"`
  InterlocutoryAttorneyName string `xml:"interlocutory-attorney-name" json:"interlocutory-attorney-name"`
  LocationCode              string `xml:"location-code" json:"location-code"`
  DayInLocation             uint   `xml:"day-in-location" json:"day-in-location"`
  StatusUpdateDate          uint   `xml:"status-update-date" json:"status-update-date"`
  StatusCode                uint   `xml:"status-code" json:"status-code"`

  PartyInformation struct {
    Party struct {
      Identifier uint   `xml:"identifier" json:"identifier"`
      RoleCode   string `xml:"role-code" json:"role-code"`
      Name       string `xml:"name" json:"name"`

      PropertyInformation struct {
        Property struct {
          Identifier   uint   `xml:"identifier" json:"identifier"`
          SerialNumber uint   `xml:"serial-number" json:"serial-number"`
          MarkText     string `xml:"mark-text" json:"mark-text"`
        } `xml:"property" json:"property"`
      } `xml:"property-information" json:"property-information"`

      AddressInformation struct {
        ProceedingAddress struct {
          Identifier uint   `xml:"identifier" json:"identifier"`
          TypeCode   string `xml:"type-code" json:"type-code"`
          Name       string `xml:"name" json:"name"`
          OrgName    string `xml:"orgname"  json:"orgname"`
          Address_1  string `xml:"address-1" json:"address-1"`
          City       string `xml:"city" json:"city"`
          State      string `xml:"state" json:"state"`
          Country    string `xml:"country" json:"country"`
          Postcode   string `xml:"postcode" json:"postcode"`
        } `xml:"proceeding-address" json:"proceeding-address"`
      } `xml:"address-information" json:"address-information"`
    } `xml:"party" json:"party"`
  } `xml:"party-information" json:"party-information"`

  ProsecutionHistory struct {
    ProsecutionEntry []ProsecutionEntry `xml:"prosecution-entry" json:"prosecution-entry"`
  } `xml:"prosecution-history" json:"prosecution-history"`
}

type ProsecutionEntry struct {
  Identifier  uint   `xml:"identifier" json:"identifier"`
  Code        uint   `xml:"code" json:"code"`
  TypeCode    string `xml:"type-code" json:"type-code"`
  Date        uint   `xml:"date" json:"date"`
  HistoryText string `xml:"history-text" json:"history-text"`
}
```

unmarshal:

```
var xml_data datatype.XmlFile
  err = xml.Unmarshal(data, &xml_data)
  if err != nil {
    log.Fatalf(err.Error())
  }
```

- Convert and export the data into JSON (intended file)

```
json_data, err := json.MarshalIndent(xml_data, "", "  ")
  if err != nil {
    log.Fatalf(err.Error())
  }

  err = os.WriteFile("DB_Result.json", json_data, 0644)
  if err != nil {
    log.Fatalf(err.Error())
  }
```

my main.go file:

```
package main

import (
  "database/sql"
  "encoding/json"
  "encoding/xml"
  "io"
  "log"
  "os"

  "github.com/technoreck/TradeMarkia_Challenge/datatype"
)

func main() {

  file, err := os.Open("tt230101.xml")
  if err != nil {
    log.Fatalf(err.Error())
  }
  defer file.Close()

  // Read the XML data
  data, err := io.ReadAll(file)
  if err != nil {
    log.Fatalf(err.Error())
  }

  var xml_data datatype.XmlFile
  err = xml.Unmarshal(data, &xml_data)
  if err != nil {
    log.Fatalf(err.Error())
  }

  json_data, err := json.MarshalIndent(xml_data, "", "  ")
  if err != nil {
    log.Fatalf(err.Error())
  }

  err = os.WriteFile("DB_Result.json", json_data, 0644)
  if err != nil {
    log.Fatalf(err.Error())
  }

}
```

- Created an instance on elephantSQL for creating a DB and connecting to it.

```
db, err := sql.Open("postgres", "postgres://irrdzyqn:rav3LEPxmu-lOBam54ikWtOdiM7BZ3QC@tiny.db.elephantsql.com/irrdzyqn")
  if err != nil {
    log.Fatalf(err.Error())
  }
  defer db.Close()

  err = db.Ping()
  if err != nil {
    log.Fatalf(err.Error())
  }
```