Files Tree Structure

|--go.sum

|--main.go

|--bin

|--folder\_to\_test

|--|--\_\_init\_\_.py

|--|--configs

|--|--|--\_\_init\_\_.py

|--|--|--base.py

|--|--constants

|--|--|--crud\_types.py

|--|--|--\_\_init\_\_.py

|--|--schemas

|--|--|--mixins.py

|--|--|--\_\_init\_\_.py

|--|--|--email\_validator.py

|--|--utilities

|--|--|--files.py

|--|--|--\_\_init\_\_.py

|--go.mod

Files Content

## main.go

package main

import (

"fmt"

"os"

"path/filepath"

"strings"

"sync"

"baliance.com/gooxml/document"

"baliance.com/gooxml/schema/soo/wml"

)

type TreeStorage struct {

Value map[string]any

mutex sync.Mutex

}

func (t \*TreeStorage) addElem(key string, value any) {

t.mutex.Lock()

t.Value[key] = value

t.mutex.Unlock()

}

func main() {

var wg sync.WaitGroup

dir, err := os.Getwd()

if err != nil {

fmt.Println("Error:", err)

return

}

fmt.Println("Current directory:", dir)

storage := &TreeStorage{Value: map[string]any{}, mutex: sync.Mutex{}}

files, \_ := os.ReadDir(dir)

for \_, file := range files {

fpath := filepath.Join(dir, file.Name())

if strings.HasPrefix(file.Name(), ".") || file.Name() == "structure.docx" || strings.HasSuffix(file.Name(), ".exe") {

continue

}

if file.IsDir() {

wg.Add(1)

go recursiveFileParser(fpath, &wg, storage)

} else {

b, err := os.ReadFile(fpath)

if err == nil {

file\_text := string(b)

storage.addElem(file.Name(), file\_text)

}

}

}

wg.Wait()

printTreeStructure("", storage.Value)

saveStructureToDocx(&storage.Value)

}

func recursiveFileParser(dir string, wg \*sync.WaitGroup, storage \*TreeStorage) {

files, \_ := os.ReadDir(dir)

res := map[string]any{}

for \_, file := range files {

if strings.HasPrefix(file.Name(), ".") || strings.HasSuffix(file.Name(), ".exe") {

continue

}

fpath := filepath.Join(dir, file.Name())

if file.IsDir() {

res[file.Name()] = recursiveSyncFileParser(fpath)

continue

}

b, err := os.ReadFile(fpath)

if err != nil {

continue

}

file\_text := string(b)

res[file.Name()] = file\_text

}

storage.addElem(filepath.Base(dir), res)

wg.Done()

}

func recursiveSyncFileParser(dir string) map[string]any {

files, \_ := os.ReadDir(dir)

res := map[string]any{}

for \_, file := range files {

fpath := filepath.Join(dir, file.Name())

if file.IsDir() {

res[file.Name()] = recursiveSyncFileParser(fpath)

continue

}

b, err := os.ReadFile(fpath)

if err != nil {

continue

}

file\_text := string(b)

res[file.Name()] = file\_text

}

return res

}

func printTreeStructure(prefix string, tree map[string]any) {

p := prefix + "|--"

for k, v := range tree {

fmt.Printf("%s%s\n", p, k)

value, ok := v.(map[string]any)

if ok {

printTreeStructure(p, value)

}

}

}

func saveStructureToDocx(tree \*map[string]any) {

doc := document.New()

para := doc.AddParagraph()

run := para.AddRun()

para.SetStyle("Title")

run.AddText("Files Tree Structure")

addTreeStructure("", tree, doc)

doc.AddParagraph().Properties().AddSection(wml.ST\_SectionMarkNextPage)

para = doc.AddParagraph()

para.SetStyle("Title")

para.AddRun().AddText("Files Content")

addFilesContent("", tree, doc)

doc.SaveToFile("structure.docx")

}

func addTreeStructure(prefix string, tree \*map[string]any, d \*document.Document) {

p := prefix + "|--"

for k, v := range \*tree {

s := fmt.Sprintf("%s%s\n", p, k)

para := d.AddParagraph()

run := para.AddRun()

run.AddText(s)

value, ok := v.(map[string]any)

if ok {

addTreeStructure(p, &value, d)

}

}

}

func addFilesContent(name\_prefix string, tree \*map[string]any, d \*document.Document) {

for k, v := range \*tree {

var title string

if name\_prefix != "" {

title = fmt.Sprintf("%s/%s\n", name\_prefix, k)

} else {

title = fmt.Sprintf("%s\n", k)

}

para := d.AddParagraph()

para.Properties().SetHeadingLevel(2)

para.AddRun().AddText(title)

value, ok := v.(map[string]any)

if ok {

addFilesContent(title, &value, d)

continue

}

lines := strings.Split(v.(string), "\n")

for \_, line := range lines {

para := d.AddParagraph()

run := para.AddRun()

run.AddText(line)

run.Properties().SetFontFamily("Courier New")

run.Properties().SetSize(9)

}

}

}

## bin

## folder\_to\_test

## folder\_to\_test /schemas

## folder\_to\_test /schemas /\_\_init\_\_.py

## folder\_to\_test /schemas /email\_validator.py

from pydantic import EmailStr

from pydantic.networks import validate\_email

from pydantic\_core import core\_schema

class EmailStrLower(EmailStr):

@classmethod

def \_validate(

cls, \_\_input\_value: str, \_: core\_schema.ValidationInfo # noqa: PYI063

) -> str:

return validate\_email(\_\_input\_value)[1].lower()

## folder\_to\_test /schemas /mixins.py

import json

from typing import Any

from pydantic import BaseModel, model\_validator

class ParseFromJsonMixin(BaseModel):

@model\_validator(mode="before")

@classmethod

def parse\_from\_json(cls, value: Any) -> Any:

if isinstance(value, str):

return json.loads(value)

return value

## folder\_to\_test /utilities

## folder\_to\_test /utilities /\_\_init\_\_.py

## folder\_to\_test /utilities /files.py

from typing import List

from fastapi import UploadFile

async def get\_names\_with\_files(files: List[UploadFile]) -> dict:

if files:

return {file.filename: file for file in files if file.filename}

return {}

## folder\_to\_test /\_\_init\_\_.py

## folder\_to\_test /configs

## folder\_to\_test /configs /\_\_init\_\_.py

## folder\_to\_test /configs /base.py

from pathlib import Path

from dotenv import load\_dotenv

from pydantic\_settings import BaseSettings

load\_dotenv()

BASE\_DIR = Path(\_\_file\_\_).parent.parent

class BaseSetting(BaseSettings):

class Config:

env\_file\_encoding = "UTF-8"

extra = "allow"

## folder\_to\_test /constants

## folder\_to\_test /constants /\_\_init\_\_.py

## folder\_to\_test /constants /crud\_types.py

from typing import TypeVar

from pydantic import BaseModel

from sqlalchemy.orm.decl\_api import DeclarativeBase

ModelType = TypeVar("ModelType", bound=DeclarativeBase)

CreateSchemaType = TypeVar("CreateSchemaType", bound=BaseModel)

UpdateSchemaType = TypeVar("UpdateSchemaType", bound=BaseModel)

## go.mod

module main

go 1.23.3

require baliance.com/gooxml v1.0.1

## go.sum

baliance.com/gooxml v1.0.1 h1:fG5lmxmjEVFfbKQ2NuyCuU3hMuuOb5avh5a38SZNO1o=

baliance.com/gooxml v1.0.1/go.mod h1:+gpUgmkAF4zCtwOFPNRLDAvpVRWoKs5EeQTSv/HYFnw=