文档

https://pkg.go.dev/go.uber.org/zap#section-readme

项目应用

使用的包:
"go.uber.org/zap"
"go.uber.org/zap/zapcore"
"github.com/natefinch/lumberjack"-- 用于编写Zap中的writer, 日志切割删除

核心函数:
zapcore.NewCore(getEncoder(), writer, level)
需要一个encoder编码器, writer,写入器, level表示日志级别

```go
package core

import (
	"fmt"
	"os"
	"time"

	"github.com/flipped-aurora/gin-vue-admin/server/global"
	"github.com/flipped-aurora/gin-vue-admin/server/utils"
	"go.uber.org/zap"
	"go.uber.org/zap/zapcore"
)

func Zap() (logger *zap.Logger) {
	if ok, _ := utils.PathExists(global.GVA_CONFIG.Zap.Director); !ok { // 判断是否有Director文件夹
		fmt.Printf("create %v directory\n", global.GVA_CONFIG.Zap.Director)
		_ = os.Mkdir(global.GVA_CONFIG.Zap.Director, os.ModePerm)
	}
	// 调试级别
	debugPriority := zap.LevelEnablerFunc(func(lev zapcore.Level) bool {
		return lev == zap.DebugLevel
	})
	// 日志级别
	infoPriority := zap.LevelEnablerFunc(func(lev zapcore.Level) bool {
		return lev == zap.InfoLevel
	})
	// 警告级别
	warnPriority := zap.LevelEnablerFunc(func(lev zapcore.Level) bool {
		return lev == zap.WarnLevel
	})
	// 错误级别
	errorPriority := zap.LevelEnablerFunc(func(lev zapcore.Level) bool {
		return lev >= zap.ErrorLevel
	})

	cores := [...]zapcore.Core{
		getEncoderCore(fmt.Sprintf("./%s/server_debug.log", global.GVA_CONFIG.Zap.Director), debugPriority),
		getEncoderCore(fmt.Sprintf("./%s/server_info.log", global.GVA_CONFIG.Zap.Director), infoPriority),
		getEncoderCore(fmt.Sprintf("./%s/server_warn.log", global.GVA_CONFIG.Zap.Director), warnPriority),
		getEncoderCore(fmt.Sprintf("./%s/server_error.log", global.GVA_CONFIG.Zap.Director), errorPriority),
	}
	logger = zap.New(zapcore.NewTee(cores[:]...), zap.AddCaller())

	if global.GVA_CONFIG.Zap.ShowLine {
		logger = logger.WithOptions(zap.AddCaller())
	}
	return logger
}

// getEncoderConfig 获取zapcore.EncoderConfig
func getEncoderConfig() (config zapcore.EncoderConfig) {
	config = zapcore.EncoderConfig{
		MessageKey:     "message",
		LevelKey:       "level",
		TimeKey:        "time",
		NameKey:        "logger",
		CallerKey:      "caller",
		StacktraceKey:  global.GVA_CONFIG.Zap.StacktraceKey,
		LineEnding:     zapcore.DefaultLineEnding,
		EncodeLevel:    zapcore.LowercaseLevelEncoder,
		EncodeTime:     CustomTimeEncoder,
		EncodeDuration: zapcore.SecondsDurationEncoder,
		EncodeCaller:   zapcore.FullCallerEncoder,
	}
	switch {
	case global.GVA_CONFIG.Zap.EncodeLevel == "LowercaseLevelEncoder": // 小写编码器(默认)
		config.EncodeLevel = zapcore.LowercaseLevelEncoder
	case global.GVA_CONFIG.Zap.EncodeLevel == "LowercaseColorLevelEncoder": // 小写编码器带颜色
		config.EncodeLevel = zapcore.LowercaseColorLevelEncoder
	case global.GVA_CONFIG.Zap.EncodeLevel == "CapitalLevelEncoder": // 大写编码器
		config.EncodeLevel = zapcore.CapitalLevelEncoder
	case global.GVA_CONFIG.Zap.EncodeLevel == "CapitalColorLevelEncoder": // 大写编码器带颜色
		config.EncodeLevel = zapcore.CapitalColorLevelEncoder
	default:
		config.EncodeLevel = zapcore.LowercaseLevelEncoder
	}
	return config
}

// getEncoder 获取zapcore.Encoder
func getEncoder() zapcore.Encoder {
	if global.GVA_CONFIG.Zap.Format == "json" {
		return zapcore.NewJSONEncoder(getEncoderConfig())
	}
	return zapcore.NewConsoleEncoder(getEncoderConfig())
}

// getEncoderCore 获取Encoder的zapcore.Core
func getEncoderCore(fileName string, level zapcore.LevelEnabler) (core zapcore.Core) {
	writer := utils.GetWriteSyncer(fileName) // 使用file-rotatelogs进行日志分割
	// 调用生成自定义ZapCore
	return zapcore.NewCore(getEncoder(), writer, level)
}

// 自定义日志输出时间格式
func CustomTimeEncoder(t time.Time, enc zapcore.PrimitiveArrayEncoder) {
    enc.AppendString(t.Format(global.GVA_CONFIG.Zap.Prefix + "2006/01/02 - 15:04:05.000"))
}

```

utils.GetWriteSyncer(filename)部分

```go
package utils

import (
    "os"

    "github.com/flipped-aurora/gin-vue-admin/server/global"
    "github.com/natefinch/lumberjack"
    "go.uber.org/zap/zapcore"
)

//@author: [SliverHorn](https://github.com/SliverHorn)
//@function: GetWriteSyncer
//@description: zap logger中加入file-rotatelogs
//@return: zapcore.WriteSyncer, error

func GetWriteSyncer(file string) zapcore.WriteSyncer {
    lumberJackLogger := &lumberjack.Logger{
        Filename:   file, // 日志文件的位置
        MaxSize:    10,   // 在进行切割之前，日志文件的最大大小（以MB为单位）
        MaxBackups: 200,  // 保留旧文件的最大个数
        MaxAge:     30,   // 保留旧文件的最大天数
        Compress:   true, // 是否压缩/归档旧文件
    }

    if global.GVA_CONFIG.Zap.LogInConsole {
        // 两个Writer
        return zapcore.NewMultiWriteSyncer(zapcore.AddSync(os.Stdout), zapcore.AddSync(lumberJackLogger))
    }
    return zapcore.AddSync(lumberJackLogger)
}

```

替代原来Log

```go
zap.ReplaceGlobals(global.GVA_LOG) // 替换zap包中全局的logger实例，后续在其他包中只需使用zap.L()调用即可
```