Урок Спринт 1.13

Zap

Zap — это быстрая и эффективная библиотека логирования для Go от `Uber`. Её преимущества перед стандартным пакетом `log` — те же, что и у `Logrus`. Кроме того, она может похвастаться высокой производительностью, за счёт чего отлично подходит для программ с большим объёмом логов.

Вот как с помощью Zap можно записать сообщение `Hello, World!` в лог уровня `Info`:

```
package main

import (
    "go.uber.org/zap"
)

func main() {
    logger, _ := zap.NewProduction()
    defer logger.Sync()

    logger.Info("Hello, World!")
}
```

При запуске этой программы в консоли появится сообщение:

```
{"level":"info","ts":1634740800.000001,"caller":"main.go:10","msg":"Hello, World!"}
```

Для разработки мы рекомендуем использовать конфигурацию логгера `Development`. Она выводит логи в удобном для чтения виде, а также включает в себя стек вызовов для каждого сообщения лога.

Вот как можно использовать этот логгер:

```
logger, err := zap.NewDevelopment()
if err != nil {
    log.Fatalf("can't initialize zap logger: %v", err)
}
defer logger.Sync()

logger.Info("Hello, world!")
```

С `Development` сообщения логов будут выводиться в таком виде:

```
2021-11-04T12:00:00.000+02:00    INFO    example/main.go:14    Hello, world!
```

Для продакшна мы рекомендуем использовать конфигурацию логгера `Production`. Она оптимизирована, чтобы быть максимально производительной и безопасной.

Вот как можно использовать этот логгер:

```
logger, err := zap.NewProduction()
if err != nil {
    log.Fatalf("can't initialize zap logger: %v", err)
}
defer logger.Sync()

logger.Info("Hello, world!")
```

С `Production` сообщения логов будут выводиться в таком виде:

```
{"level":"info","ts":1636046400.000001,"caller":"example/main.go:14","msg":"Hello, world!"}
```

Конфигурация	Печатает stack trace для	Печатает файл/номер строки	Печатает уровень сообщения в	Печатает время в формате
Development	Warn и выше	всегда	верхнем регистре	ISO8601 с миллисекундами
Production	Error/DPanic levels	всегда	нижнем регистре	epoch

В `Production`-конфигурации также добавляется поле "`caller`", которое указывает, в каком месте был написан лог.

Среди прочего, `Zap` включает функцию `WithFields`, которая добавляет поля (`fields`) к логам. Там может содержаться дополнительная информация о событиях или контексте. Это делает логирование информативнее и полезнее для анализа.

Вот как можно использовать функцию `WithFields` в `Zap`:

```
package main

import (
    "go.uber.org/zap"
)

func main() {
    // Создаём конфигурацию для логгера
    logger, err := zap.NewProduction()
    defer logger.Sync()

    // Добавляем поля
    logger.WithFields(zap.String("user", "JohnDoe"), zap.Int("age", 30)).
        Info("User login")

    // Другой пример с полями
    logger.WithFields(zap.String("city", "New York"), zap.Float64("temperature", 78.5)).
        Warn("Weather update")

    // Логируем, не добавляя поля
    logger.Info("Simple log message")
}
```

```
// Вывод в консоль
{"level":"info","ts":1694754718.482332,"caller":"awesomeProject14/main.go:15","msg":"User login","user":"JohnDoe","age":30}
{"level":"warn","ts":1694754718.4825048,"caller":"awesomeProject14/main.go:19","msg":"Weather update","city":"New York","temperature":78.5
{"level":"info","ts":1694754718.4825149,"caller":"awesomeProject14/main.go:22","msg":"Simple log message"}
```

В Zap есть два основных `API` для ведения логов:

- **Logger** — низкоуровневый тип. Обеспечивает структурированный способ написания сообщений. Поддерживает только строго типизированные контекстные поля. Подходит, если вам нужна высокая производительность программы
- **SugaredLogger** — тип высокого уровня. Затратнее по ресурсам, чем `Logger`

```
package main

import (
    "go.uber.org/zap"
    "time"
)

func main() {
    logger, _ := zap.NewProduction()
    defer logger.Sync() // flushes buffer, if any
    // Вызываем SugaredLogger
    sugar := logger.Sugar()
    sugar.Infow("failed to fetch URL",
        // Можем прокидывать любые типы и не указывать их
        "url", "какой-то url",
        "attempt", 3,
        "backoff", time.Second,
    )
    sugar.Infof("Failed to fetch URL: %s", "какой-то url")
}
```

Подробнее об `API` можно прочесть в **документации самого** `zap`.

Справка