

# GO-06 06: Профилирование с pprof

## Описание:

Команды

Запуск бенчмарка с сохранением профилей CPU и MEM.

```
go test -bench=. -benchtime=5s -cpuprofile=cpu.profile  
-memprofile=mem.profile
```

Запуск go tool pprof для просмотра профайлов.

```
go tool pprof -http=:9090 [profile_file]
```

Запуск go tool pprof на работающем сервисе для анализа работы CPU.

```
go tool pprof -http=:9090  
http://host:port/debug/pprof/profile?seconds=5
```

Запуск go tool pprof на работающем сервисе для анализа работы памяти.

```
go tool pprof -http=:9090 http://host:port/debug/pprof/allocs  
go tool pprof -http=:9090 http://host:port/debug/pprof/heap
```

Запуск go tool pprof на работающем сервисе для анализа асинхронной работы.

```
go tool pprof -http=:9090 http://host:port/debug/pprof/goroutine  
go tool pprof -http=:9090 http://host:port/debug/pprof/mutex  
go tool pprof -http=:9090 http://host:port/debug/pprof/threadcreate  
go tool pprof -http=:9090 http://host:port/debug/pprof/block
```

Запуск go tool trace на работающем сервисе.

```
curl http://host:port/debug/pprof/trace?seconds=10 -o trace.out  
go tool trace -http:9191 ./trace.out
```

## Полезные ссылки:

- [Профилирование и оптимизация программ на Go](#)
- [Профилирование и оптимизация веб-приложений на Go](#)
- [PPROF Github](#)
- [An Introduction to Go Tool Trace](#)

## Задание:

В этом задании нам потребуется посмотреть на работу нашего сервиса со стороны производительности и попробовать внести изменения для его улучшения. API эндпоинт `/hello` генерирует большое количество сообщений `Hello, world` разной длины. Генерация находится в функции `Pad` в пакете `./internal/pkg/util`.

1. В вашем проекте `module06` сделайте новую ветку `module06_06`.
2. Запустите сервис (`main` файл сервиса находится в каталоге `./cmd/app/main.go`).
3. Сгенерируйте нагрузку на наш сервис одним из нескольких способов:
  - через `curl while ; do curl http://host:port/hello; sleep 1; done;`
  - через утилиты нагрузочного тестирования такие, как `wrk`, `ab` и т.д.
4. Проанализируйте работу CPU и MEM непосредственно на работающем сервисе.
5. Попробуйте улучшить результат.
6. Снова проанализируйте работу CPU и MEM, но уже с оптимизированным вариантом.
7. В качестве ответа пришлите ссылку на `merge request` в ветку `master` вашего проекта ветки `module06_06`, в которой должны быть:
  - скриншоты из `rrprof` с профайлами CPU\MEM до оптимизации и после;
  - оптимизированный вариант функции `Pad` (подумайте над использованием `strings.Builder`).