Go gRPC Middleware

gRPC Go Middleware: interceptors, helpers, utilities.

Middleware

gRPC Go recently acquired support for Interceptors, i.e. middleware that is executed either on the gRPC Server before the request is passed onto the user's application logic, or on the gRPC client around the user call. It is a perfect way to implement common patterns: auth, logging, message, validation, retries or monitoring.

These are generic building blocks that make it easy to build multiple microservices easily. The purpose of this repository is to act as a go-to point for such reusable functionality. It contains some of them itself, but also will link to useful external repos.

`grpc_middleware` itself provides support for chaining interceptors, here's an example:

```
import "github.com/grpc-ecosystem/go-grpc-middleware"

myServer := grpc.NewServer(
    grpc.StreamInterceptor(grpc_middleware.ChainStreamServer(
        grpc_recovery.StreamServerInterceptor(),
        grpc_ctxtags.StreamServerInterceptor(),
        grpc_opentracing.StreamServerInterceptor(),
        grpc_prometheus.StreamServerInterceptor,
        grpc_zap.StreamServerInterceptor(zapLogger),
        grpc_auth.StreamServerInterceptor(myAuthFunction),
    )),
    grpc.UnaryInterceptor(grpc_middleware.ChainUnaryServer(
        grpc_recovery.UnaryServerInterceptor(),
        grpc_ctxtags.UnaryServerInterceptor(),
        grpc_opentracing.UnaryServerInterceptor(),
        grpc_prometheus.UnaryServerInterceptor,
        grpc_zap.UnaryServerInterceptor(zapLogger),
        grpc_auth.UnaryServerInterceptor(myAuthFunction),
    )),
)
```

Interceptors

Please send a PR to add new interceptors or middleware to this list

Auth

- `grpc_auth` - a customizable (via `AuthFunc`) piece of auth middleware

Logging

- `grpc_ctxtags` - a library that adds a `Tag` map to context, with data populated from request body
- `grpc_zap` - integration of zap logging library into gRPC handlers.
- `grpc_logrus` - integration of logrus logging library into gRPC handlers.
- `grpc_kit` - integration of go-kit logging library into gRPC handlers.
- `grpc_grpc_logsettable` - a wrapper around `grpclog.LoggerV2` that allows to replace loggers in runtime (thread-safe).

Monitoring

- `grpc_prometheus` ⚡ - Prometheus client-side and server-side monitoring middleware
- `otgrpc` ⚡ - OpenTracing client-side and server-side interceptors
- `grpc_opentracing` - OpenTracing client-side and server-side interceptors with support for streaming and handler-returned tags

Client

- `grpc_retry` - a generic gRPC response code retry mechanism, client-side middleware

Server

- `grpc_validator` - codegen inbound message validation from `.proto` options
- `grpc_recovery` - turn panics into gRPC errors
- `ratelimit` - grpc rate limiting by your own limiter

Status

This code has been running in *production* since May 2016 as the basis of the gRPC micro services stack at Improbable.

Additional tooling will be added, and contributions are welcome.

License

`go-grpc-middleware` is released under the Apache 2.0 license. See the LICENSE file for details.