

GoFrame3

 OpenTelemetry attributes Prometheus labels

Content Menu

-
-
-

MeterMeterMeter

```
package main

import (
    "context"

    "go.opentelemetry.io/otel/exporters/prometheus"

    "github.com/gofr/gf/contrib/metric/otelmetric/v2"
    "github.com/gofr/gf/v2/frame/g"
    "github.com/gofr/gf/v2/os/gctx"
    "github.com/gofr/gf/v2/os/gmetric"
)

const (
    instrument      = "github.com/gofr/gf/example/metric/basic"
    instrumentVersion = "v1.0"
)

var (
    meter = gmetric.GetGlobalProvider().Meter(gmetric.MeterOption{
        Instrument:      instrument,
        InstrumentVersion: instrumentVersion,
        Attributes: gmetric.Attributes{
            gmetric.NewAttribute("meter_const_attr_1", 1),
        },
    })
    counter = meter.MustCounter(
        "goframe.metric.demo.counter",
        gmetric.MetricOption{
            Help: "This is a simple demo for Counter usage",
            Unit: "bytes",
            Attributes: gmetric.Attributes{
                gmetric.NewAttribute
            ("metric_const_attr_1", 1),
            },
        },
    )
    observableCounter = meter.MustObservableCounter(
        "goframe.metric.demo.observable_counter",
        gmetric.MetricOption{
            Help: "This is a simple demo for ObservableCounter
usage",
            Unit: "%",
            Attributes: gmetric.Attributes{
                gmetric.NewAttribute
            ("metric_const_attr_2", 2),
            },
        },
    )
)

func main() {
    var ctx = gctx.New()
    // Callback for observable metrics.
    meter.MustRegisterCallback(func(ctx context.Context, obs gmetric.
Observer) error {
```

```
        obs.Observe(observableCounter, 10)
        return nil
    }, observableCounter)

    // Prometheus exporter to export metrics as Prometheus format
    exporter, err := prometheus.New(
        prometheus.WithoutCounterSuffixes(),
        prometheus.WithoutUnits(),
    )
    if err != nil {
        g.Log().Fatal(ctx, err)
    }

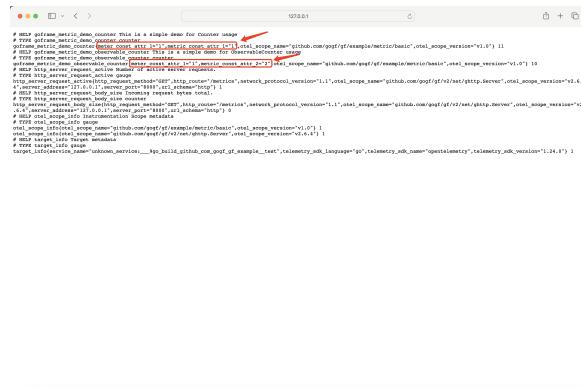
    // OpenTelemetry provider.
    provider := otelmetric.MustProvider(
        otelmetric.WithReader(exporter),
    )
    provider.SetAsGlobal()
    defer provider.Shutdown(ctx)

    // Counter.
    counter.Inc(ctx)
    counter.Add(ctx, 10)

    // HTTP Server for metrics exporting.
    otelmetric.StartPrometheusMetricsServer(8000, "/metrics")
}
```

MeterMetricMeterOptionMetricOptionAttributes

<http://127.0.0.1:8000/metrics> Meter



```
package main

import (
    "context"

    "go.opentelemetry.io/otel/exporters/prometheus"

    "github.com/gogf/gf/contrib/metric/otelmetric/v2"
    "github.com/gogf/gf/v2/frame/g"
    "github.com/gogf/gf/v2/os/gctx"
    "github.com/gogf/gf/v2/os/gmetric"
)

const (
```

```

instrument      = "github.com/gogf/gf/example/metric/basic"
instrumentVersion = "v1.0"
}

var (
    meter = gmetric.GetGlobalProvider().Meter(gmetric.MeterOption{
        Instrument:           instrument,
        InstrumentVersion:   instrumentVersion,
        Attributes:          gmetric.Attributes{
            gmetric.NewAttribute("meter_const_attr_1", 1),
        },
    })
    counter = meter.MustCounter(
        "goframe.metric.demo.counter",
        gmetric.MetricOption{
            Help: "This is a simple demo for Counter usage",
            Unit: "bytes",
            Attributes: gmetric.Attributes{
                gmetric.NewAttribute
("metric_const_attr_1", 1),
            },
        },
    )
    observableCounter = meter.MustObservableCounter(
        "goframe.metric.demo.observable_counter",
        gmetric.MetricOption{
            Help: "This is a simple demo for ObservableCounter
usage",
            Unit: "%",
            Attributes: gmetric.Attributes{
                gmetric.NewAttribute
("metric_const_attr_2", 2),
            },
        },
    )
)

func main() {
    var ctx = gctx.New()
    // Callback for observable metrics.
    meter.MustRegisterCallback(func(ctx context.Context, obs gmetric.
Observer) error {
        obs.Observe(observableCounter, 10, gmetric.Option{
            Attributes: gmetric.Attributes{
                gmetric.NewAttribute("dynamic_attr_1", 1),
            },
        })
        return nil
    }, observableCounter)

    // Prometheus exporter to export metrics as Prometheus format.
    exporter, err := prometheus.New(
        prometheus.WithoutCounterSuffixes(),
        prometheus.WithoutUnits(),
    )
    if err != nil {
        g.Log().Fatal(ctx, err)
    }

    // OpenTelemetry provider.
    provider := otelmetric.MustProvider(
        otelmetric.WithReader(exporter),
    )
    provider.SetAsGlobal()
    defer provider.Shutdown(ctx)

    // Counter.
    counter.Inc(ctx, gmetric.Option{
        Attributes: gmetric.Attributes{
            gmetric.NewAttribute("dynamic_attr_2", 2),
        },
    })
}

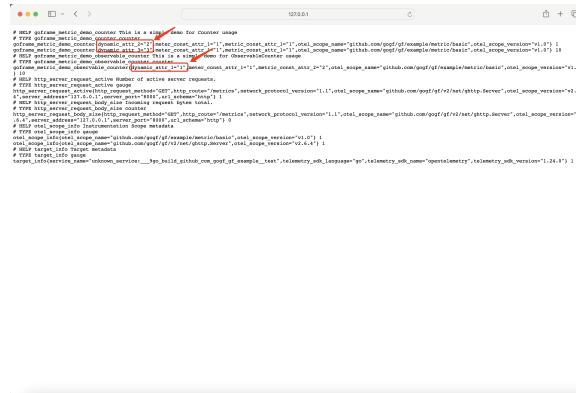
```

```
        })
    counter.Add(ctx, 10, gmetric.Option{
        Attributes: gmetric.Attributes{
            gmetric.NewAttribute("dynamic_attr_3", 3)
        },
    })
}

// HTTP Server for metrics exporting.
otelmetric.StartPrometheusMetricsServer(8000, "/metrics")
}
```

OptionAttributes

<http://127.0.0.1:8000/metrics>



Instrument

```
package main

import (
    "context"

    "go.opentelemetry.io/otel/exporters/prometheus"

    "github.com/gogf/gf/contrib/metric/otelmetric/v2"
    "github.com/gogf/gf/v2/frame/g"
    "github.com/gogf/gf/v2/os/gctx"
    "github.com/gogf/gf/v2/os/gmetric"
)

const (
    instrument          = "github.com/gogf/gf/example/metric/basic"
    instrumentVersion = "v1.0"
)

var (
    meter = gmetric.GetGlobalProvider().Meter(gmetric.MeterOption{
        Instrument:          instrument,
        InstrumentVersion: instrumentVersion,
        Attributes: gmetric.Attributes{
            gmetric.NewAttribute("meter_const_attr_1", 1),
        },
    })
    counter = meter.MustCounter(
        "goframe.metric.demo.counter",
        gmetric.MetricOption{
            Help: "This is a simple demo for Counter usage",
            Unit: "bytes",
        },
    )
)
```

```

        Attributes: gmetric.Attributes{
            gmetric.NewAttribute
("metric_const_attr_1", 1),
        },
    },
)
observableCounter = meter.MustObservableCounter(
    "goframe.metric.demo.observable_counter",
    gmetric.MetricOption{
        Help: "This is a simple demo for ObservableCounter
usage",
        Unit: "%",
        Attributes: gmetric.Attributes{
            gmetric.NewAttribute
("metric_const_attr_2", 2),
        },
    },
)
}

func main() {
    var ctx = gctx.New()

    gmetric.SetGlobalAttributes(gmetric.Attributes{
        gmetric.NewAttribute("global_attr_1", 1),
    }, gmetric.SetGlobalAttributesOption{
        Instrument:           instrument,
        InstrumentVersion:   instrumentVersion,
        InstrumentPattern:   "",
    })

    // Callback for observable metrics.
    meter.MustRegisterCallback(func(ctx context.Context, obs gmetric.
Observer) error {
        obs.Observe(observableCounter, 10, gmetric.Option{
            Attributes: gmetric.Attributes{
                gmetric.NewAttribute("dynamic_attr_1", 1),
            },
        })
        return nil
    }, observableCounter)

    // Prometheus exporter to export metrics as Prometheus format.
    exporter, err := prometheus.New(
        prometheus.WithoutCounterSuffixes(),
        prometheus.WithoutUnits(),
    )
    if err != nil {
        g.Log().Fatal(ctx, err)
    }

    // OpenTelemetry provider.
    provider := otelmetric.MustProvider(
        otelmetric.WithReader(exporter),
    )
    provider.SetAsGlobal()
    defer provider.Shutdown(ctx)

    // Counter.
    counter.Inc(ctx, gmetric.Option{
        Attributes: gmetric.Attributes{
            gmetric.NewAttribute("dynamic_attr_2", 2),
        },
    })
    counter.Add(ctx, 10, gmetric.Option{
        Attributes: gmetric.Attributes{
            gmetric.NewAttribute("dynamic_attr_3", 3),
        },
    })
}

// HTTP Server for metrics exporting.

```

```
        otelmetric.StartPrometheusMetricsServer(8000, "/metrics")
    }
```

```
gmetric.SetGlobalAttributes(gmetric.SetGlobalAttributesOption
```

```
http://127.0.0.1:8000/metrics
```



```
# HELP gitname_metric_desc This is a simple desc for function usage
# TYPE gitname_metric_desc counter
gitname_metric_desc{count=1} 1
# HELP metric_desc_count This is a simple desc for function usage
# TYPE metric_desc_count counter
metric_desc_count{desc="example metric basic", scope="basic"} 1
# HELP otel_scope_name observable counter This is a simple desc for observable counter scope
# TYPE otel_scope_name counter
otel_scope_name{scope="basic"} 1
# HELP otel_scope_version observable counter This is a simple desc for observable counter scope
# TYPE otel_scope_version counter
otel_scope_version{scope="basic"} 1
# HELP http_server_requests_active number of active server requests
http_server_requests_active{method="GET", http_code="200", host="127.0.0.1", interval="1m", otel_scope_name="github.com/golang/example/hello", otel_scope_version="v2.4", otel_scope_attributes="{}"} 1
# HELP http_server_requests_active_exceptions_total total number of active server requests with errors
http_server_requests_active_exceptions_total{method="GET", http_code="500", host="127.0.0.1", interval="1m", otel_scope_name="github.com/golang/example/hello", otel_scope_version="v2.4", otel_scope_attributes="{}"} 1
# HELP http_server_requests_body_size_bytes total bytes of body size
http_server_requests_body_size_bytes{method="GET", http_code="200", host="127.0.0.1", interval="1m", otel_scope_name="github.com/golang/example/hello", otel_scope_version="v2.4", otel_scope_attributes="{}"} 1
# HELP http_server_requests_duration_ms duration of requests in milliseconds
http_server_requests_duration_ms{method="GET", http_code="200", host="127.0.0.1", interval="1m", otel_scope_name="github.com/golang/example/hello", otel_scope_version="v2.4", otel_scope_attributes="{}"} 1
# HELP http_server_requests_exceptions_total total number of errors
http_server_requests_exceptions_total{method="GET", http_code="500", host="127.0.0.1", interval="1m", otel_scope_name="github.com/golang/example/hello", otel_scope_version="v2.4", otel_scope_attributes="{}"} 1
# HELP http_server_requests_handling_time_ms handling time of requests in milliseconds
http_server_requests_handling_time_ms{method="GET", http_code="200", host="127.0.0.1", interval="1m", otel_scope_name="github.com/golang/example/hello", otel_scope_version="v2.4", otel_scope_attributes="{}"} 1
# HELP http_server_requests_latency_ms latency of requests in milliseconds
http_server_requests_latency_ms{method="GET", http_code="200", host="127.0.0.1", interval="1m", otel_scope_name="github.com/golang/example/hello", otel_scope_version="v2.4", otel_scope_attributes="{}"} 1
# HELP http_server_requests_processing_time_ms processing time of requests in milliseconds
http_server_requests_processing_time_ms{method="GET", http_code="200", host="127.0.0.1", interval="1m", otel_scope_name="github.com/golang/example/hello", otel_scope_version="v2.4", otel_scope_attributes="{}"} 1
# HELP http_server_requests_total total number of requests
http_server_requests_total{method="GET", http_code="200", host="127.0.0.1", interval="1m", otel_scope_name="github.com/golang/example/hello", otel_scope_version="v2.4", otel_scope_attributes="{}"} 1
# HELP target_id_for_service_renderer unknown service ...
target_id_for_service_renderer{unknown_service="..._byt_buid_github_com_gopl_if_example_test", otel_scope_name="otellemetry", otel_scope_version="1.24.3"} 1
```