MicroProfile Metrics

This cheat sheet covers the basics of MicroProfile Metrics specification uses.

DIFFERENT SCOPES OF METRICS

base metrics

`/metrics/base` / metrics that all MicroProfile vendors have to provide

vendor specific metrics

`/metrics/vendor` / vendor specific metrics (optional)

application metrics

`/metrics/application` / application-specific metrics (optional)

APPLICATION METRICS PROGRAMMING MODEL

There are two ways of registering application metrics: annotations and dynamic registration.

annotations based application metrics

`@Counted` / denotes a counter, which counts the invocations of the annotated object

`@ConcurrentGauge` / denotes a gauge which counts the parallel invocations of the annotated object

`@Gauge` / denotes a gauge, which samples the value of the annotated object

`@Metered` / denotes a meter, which tracks the frequency of invocations of the annotated object

`@Metric` / an annotation that contains the metadata information when requesting a metric to be injected or produced. This annotation can be used on fields of type `Meter`, `Timer`, `Counter`, and `Histogram`. For `Gauge`, the `@Metric` annotation can only be used on producer methods/fields

`@Timed` / denotes a timer, which tracks duration of the annotated object

`@RegistryType` / qualifies the scope of the Metric Registry to inject when injecting a `MetricRegistry`

registering metrics dynamically

To register or unregister any metric dynamically you can use the methods of `MetricRegistry`. For example:

```
@ApplicationScoped
public class MetricBean {

    private static final String COUNTER_NAME =
"test-counter";

    @Inject
    @RegistryType(type =
MetricRegistry.Type.APPLICATION)
    MetricRegistry applicationMetricRegistry;

    public void registerMetric() {
        Counter counter =
applicationMetricRegistry.counter(COUNTER_NAME);
        counter.inc();
    }

    public void unregisterMetric() {

applicationMetricRegistry.remove(COUNTER_NAME);
    }
}
```

Note: Only application scope metrics can be registered.

TAGS

Values must match `[a-zA-Z_][a-zA-Z0-9_]*` (Ascii alphabet, numbers and underscore).

defining a tag

define it at the level of a metric or define it at the level of the application server using the MicroProfile Config property `mp.metrics.tags`

Note: Values must escape equal symbols `(=)` *and commas* `(,)` *with a backslash* `(\)`.

Example:

```
mp.metrics.tags=app=shop,tier=integration,special=d
eli\=ver\,y
```

REST API

Exposed at `/metrics` endpoint. More detailed access is available at `/metrics/<scope>` (see Different scopes of metrics) or `/metrics/<scope>/<metric_name>`.

data formats

`OpenMetrics` / default, Prometheus format

`JSON` / when `application/json` is specified as HTTP `Accept` header

status codes

`200` / successful invocation

`204` / sub-tree exits but has no metrics (e.g. no application metrics defined)

`404` / in requested item does not exist

`406` / invalid HTTP Accept header

`500` / error in invocation

Author Martin Stefanko