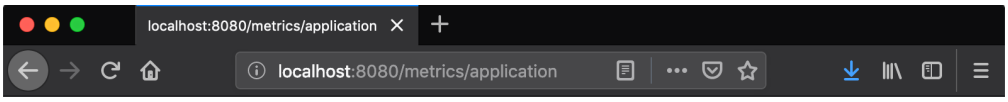
MicroProfile Metrics

When we build micro services or web applications, we need to monitor our application that it's running, have memory or disk space and for that we have MicroProfile Metrics which is very easy to get started with and use. Open the BookStoreEndpoint.java file and make the getAll method to look like this.

```
@Timed(name = "getAllBooks",
            description = "Monitor the time getAll Method takes",
            unit = MetricUnits.MILLISECONDS,
            absolute = true)
@GET
public Response getAll() {
        return Response.ok(bookService.getAll()).build();
}
```

`@Timed` annotation will monitor how long the process take. The metadata fields on `@Timed` annotation are optional, but we have added a few name field is the name of the metric, description is used to describe the metric, unit sets the unit of the metric and absolute is used to determine if the name specified in the name field is the exact name. Now Kill the Payara Micro server and start it again using the command `mvn clean package payara-micro:start` and first navigate to the http://localhost:8080/restapi/books, because we need to time it and see how long the process will take, now open another tab and go to http://localhost:8080/metrics/application and voila you have some metrics.




```
# TYPE application:get_book counter
# HELP application:get_book Monitor how many times getBook method was called
application:get_book 2
# TYPE application:create_books_total counter
# HELP application:create_books_total Monitor the rate events occured
application:create_books_total 2
# TYPE application:create_books_rate_per_second gauge
application:create_books_rate_per_second 0.0010272954851753117
# TYPE application:create_books_one_min_rate_per_second gauge
application:create_books_one_min_rate_per_second 3.1020285506000993E-13
# TYPE application:create_books_five_min_rate_per_second gauge
application:create_books_five_min_rate_per_second 2.95834602852581E-5
# TYPE application:create_books_fifteen_min_rate_per_second gauge
application:create_books_fifteen_min_rate_per_second 3.4742539510852473E-4
# TYPE application:get_all_books_rate_per_second gauge
application:get_all_books_rate_per_second 0.002568243075597919
# TYPE application:get_all_books_one_min_rate_per_second gauge
application:get_all_books_one_min_rate_per_second 2.880528958059601E-10
# TYPE application:get_all_books_five_min_rate_per_second gauge
application:get_all_books_five_min_rate_per_second 4.3993110623390927E-4
# TYPE application:get_all_books_fifteen_min_rate_per_second gauge
application:get_all_books_fifteen_min_rate_per_second 0.023986553903088032
# TYPE application:get_all_books_mean_seconds gauge
application:get_all_books_mean_seconds 1871.9183086086873
# TYPE application:get_all_books_max_seconds gauge
application:get_all_books_max_seconds 57441.92
# TYPE application:get_all_books_min_seconds gauge
application:get_all_books_min_seconds 881.682
# TYPE application:get_all_books_stddev_seconds gauge
application:get_all_books_stddev_seconds 85.52434777596713
# TYPE application:get_all_books_seconds summary
# HELP application:get_all_books_seconds Monitor the time getAll Method takes
application:get_all_books_seconds_count 5
application:get_all_books_seconds{quantile="0.5"} 1873.259
application:get_all_books_seconds{quantile="0.75"} 1873.259
application:get_all_books_seconds{quantile="0.95"} 1873.259
application:get_all_books_seconds{quantile="0.98"} 1873.259
application:get_all_books_seconds{quantile="0.99"} 1873.259
application:get_all_books_seconds{quantile="0.999"} 1873.259
```

Here we have a list of metrics and if we want a single metric then we could use the name we specified in the name field and navigate to http://localhost:8080/metrics/application/get-all-books.

@Metered

`@Metered` annotation will monitor the rate events occurred. The meta fields are optional, but it makes the life easy if we add some data to the meta fields. Change the create method to look like this.

```
@Metered(name = "create-books",
            unit = MetricUnits.MILLISECONDS,
            description = "Monitor the rate events occured",
            absolute = true)
@POST
public Response create(Book book) {
        bookService.create(book);
        return Response.ok().build();
}
```

Like the `@Timed` annotation, we have name, unit, description and absolute, which is almost identical.

@Counted

`@Counted` annotation will monitor how many times a method got invoked, and the `@Counted` annotation have a few meta fields and are optional. Update the getBook method to look like this.

```
@Counted(unit = MetricUnits.NONE,
            name = "getBook",
            absolute = true,
            monotonic = true,
            displayName = "get single book",
            description = "Monitor how many times getBook method was called")
@GET
@Path("{id}")
public Response getBook(@PathParam("id") Long id) {
        Book book = bookService.findById(id);

        return Response.ok(book).build();
}
```

Here, like the other metrics, we have name, absolute, monotonic, displayName and description, the table below show what everything is for:

unit	sets the unit of the metric.
absolute	is used to determine if the name specified in the name field is the exact name.
monotonic	is set to true, which means the counter increases monotonically.
displayName	the display name of the counter

description	describe the metric

@Gauge

`@Gauge` annotation is used to return just a value The metadata fields on @Counted annotation are optional

Example:

```
@GET
@Path("/get-int-value")
@Gauge(unit = MetricUnits.NONE, name = "intValue", absolute = true)
public int getIntValue() {
        return 3;
}
```

unit	sets the unit of the metric.
name	the name of the gauge.
absolute	is used to determine if the name specified in the name field is the exact name.

Start the application server and go to http://localhost:8080/metrics/application, you should see all your metrics.

Summary

In this chapter, we learned how to add MicroProfile Metrics to our application.