LO54

Rapport de projet

Technologie Metrics

Olivier CARDOSO

Mathilde PERROT

Info 05

Automne 2018

Sommaire

Introduction

Ce rapport présente la technologie utilisée lors du projet de développement d'une application web avec JEE pour le cours LO54.

L'application développée permet de consulter le catalogue de formations d'une école privée et de gérer les inscriptions d'étudiants aux différents cours proposés. Ce rapport se concentre sur la partie concernant l'étude d'une technologie et son ajout dans l'application. La technologie sélectionnée est ici la librairie Metrics. La tâche à effectuer est d'afficher des mesures de temps prises aux différentes couches du logiciel.

Dans une première partie, la librairie Metrics est présentée avec les différents types de mesures pouvant être effectuées et leur utilité. Un tutoriel est ensuite présenté pour décrire l'utilisation de la librairie et son intégration dans l'application. Enfin, une conclusion sur le sujet est apportée avec une description de l'expérience vécue sur l'apprentissage de la technologie et son intégration.

I. Présentation des Metrics

La librairie Metrics permet de fournir des mesures sur le comportement des différents composants d'une application. Le but est de pouvoir superviser les performances de l'application en exposant le fonctionnement de ses différentes couches sous forme de rapport de mesures.

En termes de fonctionnement (cf. Figure 1), les Metrics sont créés au sein des différents services de l'application, partout où l'on souhaite effectuer des mesures. Ils sont alors enregistrés dans un registre (MetricRegistry) qui regroupe l'ensemble des Metrics de l'application. Les Reporters sont ensuite utilisés pour consulter le registre, exporter les Metrics et les représenter sous forme de rapport. Il existe plusieurs types de Reporters (Console, CSV, Graphite, etc.) qui extraient les données vers différents supports (affichage console, sur un serveur, dans fichier externe, etc.).

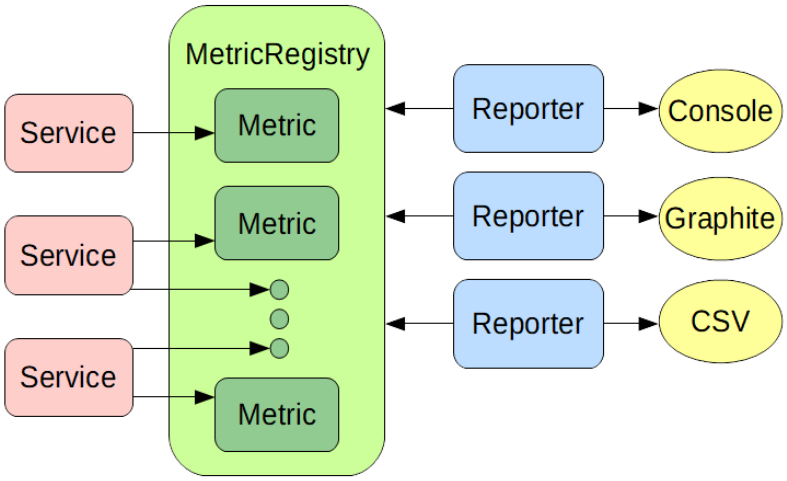

Figure 1 : Principe de fonctionnement des Metrics

Il existe plusieurs types de Metrics, chacun permettant de représenter des données statistiques concernant l'utilisation de fonctions ou de variables, les types d'opérations effectuées et leur fréquence, etc.

Meter

Un Meter permet de mesurer la fréquence à laquelle survient un événement de l'application. On l'utilise, par exemple, pour visualiser la fréquence d'appel à différentes méthodes. Le Meter fournit le taux d'occurrence par seconde de l'événement ciblé, sur 1 minute, 5 minutes, 15 minutes et sur l'ensemble de la durée de vie de l'application.

Histogram

Un Histogram permet de donner une représentation de la distribution des valeurs sur un ensemble de données. Il transmet la valeur moyenne, le minimum, le maximum, etc. Il permet également de connaître la valeur des percentiles (valeur maximum pour 75% des données, 95%, 98%, etc.).

Timer

Un Timer permet d'évaluer la vitesse de traitement d'une étape. C'est une association entre un Meter et un Histogram des durées. Il fournit donc, comme le Meter, le taux d'occurrence de l'événement par seconde sur plusieurs périodes. Il fournit également, comme l'Histogram, les temps d'exécution minimal, maximal, moyen, etc. sur l'ensemble des occurrences de l'événement ciblé.

Gauge

La Gauge est une mesure instantanée. Elle permet de monitorer l'évolution d'une valeur. Par exemple, on souhaite connaître le nombre de sessions auxquelles un étudiant est inscrit. La Gauge sera associée à la taille de la liste.

Metrics propose également un type Counter similaire à la Gauge. Il représente une mesure dont on peut incrémenter et décrémenter la valeur et s'utilise, comme son nom l'indique, pour compter des éléments.

II. Tutoriel : utilisation des Metrics

Installation

Pour installer la librairie Metrics il faut ajouter au fichier pom.xml les modules suivants :

```
<dependency>
    <groupId>io.dropwizard.metrics</groupId>
    <artifactId>metrics-core</artifactId>
    <version>4.0.2</version>
</dependency>
<dependency>
    <groupId>io.dropwizard.metrics</groupId>
    <artifactId>metrics-servlets</artifactId>
    <version>3.1.0</version>
</dependency>
```

Le premier module permet, entre autres, d'accéder aux différents types de Metrics (Meter, Timer, etc.). Le second module permet d'utiliser les Metrics dans un environnement de servlets pour les applications JEE.

Enregistrement du registre

Comme expliqué précédemment, tous les Metrics sont enregistrés dans un unique registre. Il est donc nécessaire d'en instancier un lors du démarrage de l'application et de faire en sorte qu'il soit accessible par toutes les classes. Pour cela, nous utilisons un ContextListener.

```
import com.codahale.metrics.MetricRegistry;
import com.codahale.metrics.servlets.MetricsServlet;

public class MetricsListener extends MetricsServlet.ContextListener {

    public static final MetricRegistry METRIC_REGISTRY = new MetricRegistry();

    @Override
    protected MetricRegistry getMetricRegistry() {
        return METRIC_REGISTRY;
    }
}
```

On installe ensuite la classe MetricsListener dans le fichier web.xml.

```
<listener>
    <listener-class>fr.utbm.ecoleapp.metrics.MetricsListener</listener-class>
</listener>
```

Le registre est désormais instancié lors du lancement de l'application.

Utilisation des Metrics

La description du sujet sur la technologie Metrics ne demande que d'afficher des mesures de temps. On se concentre alors sur l'utilisation du Timer. À noter que l'implémentation des autres types de Metrics reste relativement simple et suit le même principe pour l'ajout au registre.

```
private final Timer addClient = METRIC_REGISTRY.timer(name(ClientService.class,"addClient"));

public Integer addClient(Client client) {
    final Timer.Context context = addClient.time();
    try {
        ClientDao dao = new ClientDao();
        return dao.addClient(client);
    } finally {
        context.stop();
    }
}
```

La première ligne permet de créer un nouveau Timer depuis le registre. La méthode time() est utilisée pour lancer la mesure, et stop() pour l'arrêter. Ici, on test le temps d'exécution de la méthode addClient() qui correspond à l'ajout d'un client dans la base.

Visualisation des résultats via HTTP

On souhaite désormais pouvoir accéder aux mesures prises durant le cycle de vie de l'application. Le type de reporter que nous avons choisi permet de lister les Metrics sur une page HTML ; ils sont ainsi consultables directement depuis l'application.

On intègre le servlet MetricsServlet dans notre fichier web.xml. Ce servlet est proposé par le module metrics-servlets que nous avons installé précédemment et permet de sérialiser en HTTP le contenu de notre registre.

```
<servlet>
    <servlet-name>metrics</servlet-name>
    <servlet-class>com.codahale.metrics.servlets.MetricsServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>metrics</servlet-name>
    <url-pattern>/metrics/*</url-pattern>
</servlet-mapping>
```

On peut désormais accéder à nos Metrics depuis la page /metrics de notre application.

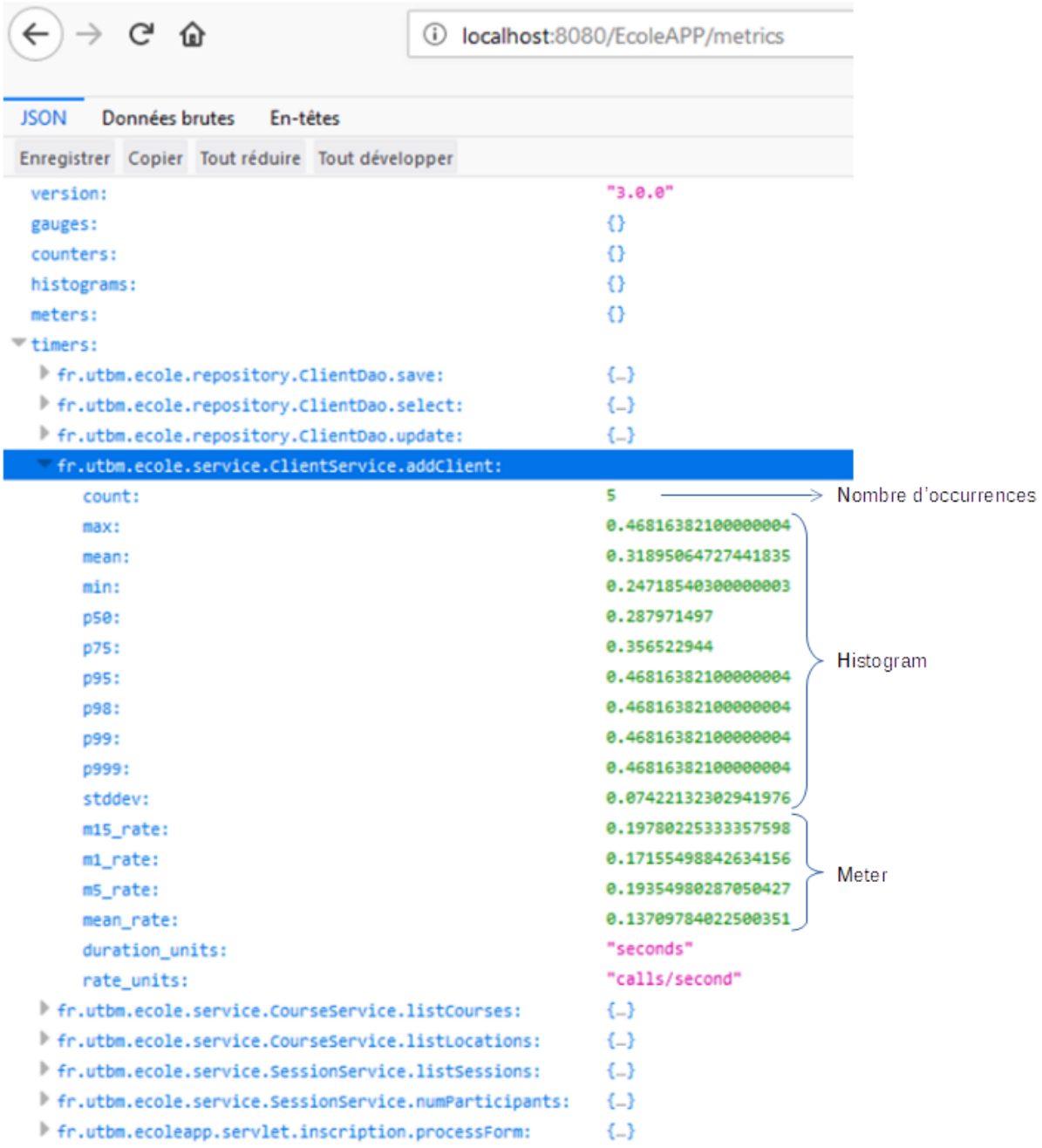

Figure 2 : Consultation des Metrics

La Figure 2 nous montre l'affichage de l'ensemble de nos Metrics. Pour chaque Timer, on peut voir dans un premier temps le nombre d'occurrences de l'événement qu'il cible durant le cycle de vie de l'application. La partie Histogram nous donne les temps d'exécutions maximum, minimum et moyen en secondes. Elle nous donne également des informations sur la distribution des temps d'exécutions avec les percentiles et l'écart type. Enfin, la partie Meter permet de connaître la fréquence d'occurrence de l'événement au cours de la dernière minute, des 5 dernières minutes, des 15 dernières minutes et de l'ensemble du cycle de vie de l'application (en appels/secondes).

Conclusion

La librairie Metrics est très utile pour cibler les parties de l'application qui manquent de performance. Cela nous permet d'identifier les étapes gourmandes en termes de temps et de nous concentrer sur la façon dont elles pourraient être optimisées. Elle permet également de comprendre plus facilement le fonctionnement de l'application et de localiser certains disfonctionnements.

L'utilisation des Metrics, de leur intégration à leur consultation, a été relativement simple à mettre en place. Leur découverte nous a permis de nous intéresser plus en détail au fonctionnement de notre application : fréquence d'utilisation des différentes fonctions durant le cycle de vie de l'application, identification des fonctions les plus consommatrices en termes de temps, découverte de certains problèmes techniques, etc.

Cette technologie nous a permis de découvrir un outil qui saura être réutilisé en milieu professionnel. Ses fonctionnalités seront appréciées durant les étapes de débuggage et d'optimisation.