Projet JEE : Rapport Technique

Roland Bavelard, Guillaume Berthe, Emmanuel Doumard, Benoît Fournier, Constance Thierry

January 13, 2017

Résumé La méthode du chaos monkey permet à un data center de tester la qualité de ses services en provoquant des pannes fréquentes sur l'infrastructure. Ce projet avait pour but de simuler le comportement du Chaos Monkey nous avons ainsi été amenés à créer deux applications, la première génère des pannes sur des machines et la seconde affiche les pannes générées. Les résultats de notre travail sont les suivants :

- Génération de pannes : normale, rafale, progressif
- Affichage des pannes par type : dernière minute, dernière heure, dernier jour, dernier mois
- Rechargement automatique partiel dans l'affichage des pannes
- Affichage sous forme de graphe du nombre de pannes par heure dans une journée

En conclusion générale, ce projet nous a permis de réaliser deux applications effectives dont le comportement simule bien celui du Chaos Monkey. De plus nous avons également eu le temps d'améliorer les applications basiques afin de les rendres plus ergonomiques.

Contents

1 Introduction

La méthode du Chaos Monkey est utilisée par certains data center afin de s'assurer qu'ils possèdent une bonne qualité de service même en cas de panne. Le principe est le suivant : générer des pannes aléatoirement sur les machines du data center afin de trouver les principales faiblesses de l'architecture existante. Notre objectif dans le cadre de ce projet est de simuler le comportement du Chaos Monkey en réalisant deux application. L'une permettant de générer des pannes, la seconde de visualiser et résoudre les pannes existantes.

2 Utilisation d'une base de données

Afin de simuler différentes machines et les différentes pannes qui peuvent leur être associées nous avons utilisé une base de données. Celle-ci contient deux tables, une table associée aux machines qui contient les informations suivantes :

Nom de la machine	Type de machine

Les machines peuvent être au choix de type : Routeur, Pare-Feux ou Serveur. Une table qui contient les informations relatives aux pannes générées:

Nom de la machine	Date de la panne	Type de la panne	État de la machine

Les pannes peuvent être de type : Reseau, Disque ou Memoire. L'état de la machine permet de savoir si la panne est toujours effective, ou si elle a été résolue.

La classe PanneDAOImpl nous permet de faire le lien entre les requêtes à la base de données et l'ensemble du projet.

3 Génération des pannes

Une première implémentation de génération de pannes consiste à utiliser le formulaire. Nous avons également eu la possibilité d'implémenter une génération de panne en mode "rafale" et en mode "progressif". Le travail le plus important dans la génération des pannes était de s'assurer que même si les paramètres entrés par l'utilisateur sont incorrects, ils n'ont pas de répercussions graves sur la base de données. Pour ce faire nous testons les attributs avant l'envoie des requêtes.

3.1 Formulaire de génération de pannes

L'utilisateur doit choisir la machine sur laquelle il souhaite lancer une panne, le type de cette panne et valider sa requête. La panne est créée via la méthode addPanne qui est proposée par PanneDAO. Cette méthode génère une requête sql qui va ajouter une panne dans le tableau relatif aux pannes dans la base de données.

3.2 Mode rafale

L'utilisateur doit renseigner le nombre X de pannes (au maximum le nombre de machines existantes) qu'il souhaite générer. La méthode addRandomPanne (présente dans la classe PanneDAO) va alors sélectionner aléatoirement X machines et les mettre en panne. Dans cette méthode on fait appel à la méthode addPanne évoquée précédement pour générer les pannes.

3.3 Mode progressif

L'utilisateur doit entrer un nombre de pannes et une durée sur laquelle va s'étendre la génération de panne. Si le nombre de pannes choisi est négatif, la fréquence de génération de pannes est

tout de même calculée, en revanche le temps de génération de pannes n'est pas borné. On génère ainsi une infinité de pannes à fréquence continue. L'implémentation du mode progressif était différente et plus complexe que les deux modes précédents et a nécessité l'utilisation d'un timer.

4 Affichage des pannes

4.1 Affichage des pannes

L'utilisateur peut visualiser les pannes qu'il a généré sur la page web d'affichage des pannes. Sur cette page on peut visualiser la liste des pannes ayant eu lieu :

- dans la dernière minute
- dans la dernière heure
- dans la dernière journée
- dans le dernier mois

L'affichage des pannes se fait celon le tableau suivant:

Nom de la machine	Type de la machine	Date de la panne	Type de la panne	Etat de la machine

Pour réaliser ce tableau nous réalisons une jointure entre nos deux tables présentes dans la base de données cette jointure est implémentée par la methode getPanneByDateDiff. Une première implémentation pour réaliser la page d'affichage des pannes a été réalisée dans un unique fichier jsp. Cependant nous avons constaté que cette implémentation n'était pas optimale lorsque nous avons commmencé à réfléchir au rechargement partiel de la page. Nous avons alors modifié notre implémentation précédente et avons décidé de créer plusieurs fichiers jsp, un pour chaque liste de pannes observés (Minute, Heure, Jour, Mois).

4.2 Rechargement de la page

Dans un premier temps nous avons implémenté le rechargement intégral de la page par l'utilisateur grâce à l'utilisation d'un bouton "refresh" (celui qui est global à la page). Il s'agissait initialement d'un classique bouton de rechargement de page, mais nous avons par la suite commencé à y intégrer les fonctions de chargement partiel. Maintenant, c'est un bouton qui initialise tout les chargements partiels. On peut discuter de la méthode, sachant que cela implique 4 appels à la base de données.

Puis nous avons dans un second temps implémenté le rechargement partiel. Celui-ci se fait par l'utilisation de boutons "refresh", un bouton est associé à une liste de panne, il recharge ainsi la liste à laquelle il est associé. Nous avons finalement implémenté un rechargement partiel de la page à intervale régulier, c'est-à-dire qu'à intervalles réguliers de temps, tout les chargements partiels sont effectués.

Pour ce qui est du fonctionnement du rechargement partiel, il est le suivant : lorsqu'une section de la page a besoin d'être rechargée, on initie un chargement sur un fichier jsp. celui-ci appel ainsi la base de données par requête sql afin de recharger les listes.

Nous n'avons pas pu implémenter le rechargement partiel sur modification de la base donnée, pour ce faire il aurait fallu utiliser un patron d'observeur-observable avec la base qui aurait déclenché le rafraichissement des JSP. La solution que nous avons trouvé était de faire de l'attente active dans ceux-ci en attendant que l'observeur leur envoie une confirmation de changement, ceux-ci se serait alors actualisés. Nous avons choisi de ne pas implémenter cette solution car elle nous semblait peu élégante, une meilleure solution doit pouvoir exister.

4.3 Affichage Graphique

Par ailleurs, à titre visuel, il a été intégré une possibilité de visualiser graphiquement les erreurs. Ceci a été fait grace au module Flot© (module gratuit de Javascript/JQuery, modèle économique par support commercial payant), qui permet d'afficher plusieurs graphes superposés.

Ce graphe fonctionne en utilisant d'une part une aquisition du temps actuel, et avec certaines méthodes de la classe PanneDAOImpl, permet d'obtenir pour une date donnée un nombre de pannes.

5 Conclusion

Nos deux applications conjointes simulent bien le comportement du Chaos Monkey. En effet, la génération de pannes est effective, de plus l'utilisateur de l'application de génération de pannes a la possibilité de choisir entre différents modes de génération. Pour la visualisation des pannes, nous avions pour objectif un système efficace permettant de visualiser plusieurs listes des pannes, en gardant une certaine ergonomie pour l'utilisateur. Notre application nous semble satisfaire ces critères : en termes d'ergonomie et de respect du cahier des charges. Nous estimons avoir répondu au critère d'efficacité par un code minimisant les charges du serveur, à l'exception de la mise à jour générale de la page.