29/01/2020

POO2 – TP4

Analyse de logs Apache

CROWLEY Sophie NASSREDDINE Zakaria
EDDAHABI Riad

Table des matières

I. Introduction

Les opérations d'accès à des sites web (requête http à un serveur, réponse du serveur, etc …) sont conservées dans un fichier journal appelé log en anglais. A travers notre application, nous souhaitons faire en sorte qu'un utilisateur puisse, à partir d'un fichier log qu'il fournit, recevoir en retour des informations sur les 10 sites les plus visités, et éventuellement la création d'un graphe grâce à l'outil GraphViz, et d'autres fonctionnalités que nous présenterons le long de ce compte rendu.

II. Spécifications

A) Spécifications générales

Entrées sur le terminal	Sortie
./main votreLog.log	On affiche les 10 documents les plus consultés
[-g votreGraphe.dot]	L'utilisateur spécifie qu'il souhaite créer un GraphViz à partir de son fichier log. On crée donc un fichier au format GraphViz
[-t heureStipulee]	L'utilisateur spécifie qu'il ne souhaite prendre en compte que les accès s'effectuant dans un intervalle caractérisé par [heureStipulee, heureStipulee+1]
[-e]	L'utilisateur spécifie qu'il souhaite exclure tous les documents de type image, javascript ou css de l'analyse du fichier log

B) Spécifications détaillées

Ligne d'exécution	Cas normaux	Cas limites
./Analog votreLog.log	- Plus de 10 documents différents sont présents dans le fichier log	- Le fichier log est vide - Le fichier n'est pas lisible - Plus de 10 documents qui peuvent faire partie du top 10 (ils ont le même nombre de hit, et c'est le maximum par exemple) - Moins de 10 documents différents consultés - Extension de fichier différente de « .log »
./Analog -e votreLog.log	- Les pages de type css, javascript et images ne sont pas prises en compte dans les logs	- Aucun document n'est d'un type autre que Js, CSS ou image - Il reste moins de 10 documents de type autre que Js, CSS ou image
./Analog -t heureStipulee votreLog.log	- Les logs en dehors de la plage horaire spécifiée ([heureStipulee, heureStipulee+1]) ne sont pas pris en compte	- Aucune page n'a été visitée durant cette plage horaire - L'heure stipulée n'est pas traitable (toute heure>24, car cela voudrait dire qu'on doit changer de jour) - Moins de 10 pages différents ont été visitées durant cette plage horaire - L'entrée n'est pas un entier
./Analog -g votreGraph.dot votreLog.log	- Un graph est créé selon le nom du fichier crée	- Nom du fichier spécifié contient des caractères non autorisés pour la création d'un fichier - Nom du fichier non spécifié

III. Architecture

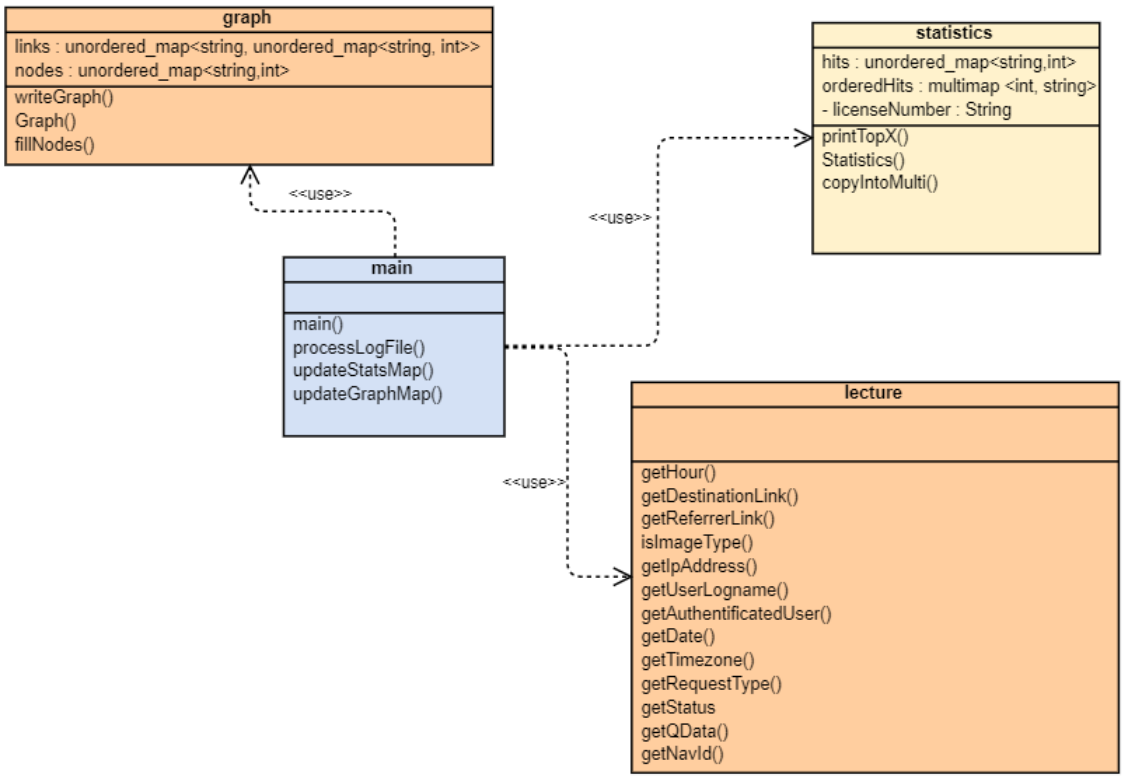

Figure 1: Diagramme UML de l'application

IV. Fonctionnement de l'application

A) Lecture des données entrées

En premier lieu, il est nécessaire à notre application d'analyser les éléments passés en arguments par l'utilisateur. Pour cela, on se sert des paramètres de la fonction main (argc et argv) qui mettent à jour des variables selon la présence ou non de certaines options dans la ligne d'exécution. Par exemple, ajouter l'option « -e » transformera la valeur du booléen « includeImages » de « true » à « false ».

B) Lecture du fichier log

Le fichier log envoyé en paramètre par l'utilisateur est lu ligne par ligne par la classe Lecture. Cette classe procède alors à un découpage des informations (selon les différentes catégories d'informations : IP, heure de consultation, …). Une méthode est assignée à chaque information. Cela permet la réutilisabilité du code pour d'autres besoins.

C) Mise en place du TOP 10

La mise en place du top 10 se fait via la classe 'Statistics'. Cette classe a deux attributs :

- hits : une unordered_map avec des clés de type string indiquant les noms des pages et des valeurs de type int indiquant le nombre de hits pour chaque page. Cette map stocke donc, pour chaque page, le nombre de hits associés. Le contenu de cette map est par la suite "transféré" vers le deuxième attribut orderedHits.

- orderedHits: il s'agit ici d'une multimap où on saisit le contenu de la map hits mais en inversant l'ordre clé-valeur, la clé devient donc le nombre de hits et la valeur le nom de la page. On a opté pour cette solution d'abord pour faciliter le tri des pages par nombre de hits, mais également pour prendre en compte le cas où des pages différentes ont le même nombre de hits.

Cette classe est également dotée de deux méthodes :

- printTopX : qui prend en entrée un paramètre entier x et affiche sur la sortie standard les x première pages classées selon leurs nombres de hits.

- copyIntoMulti : cette méthode assure le transfert des données contenues dans la map hits vers la multimap orderedHits pour pouvoir trier les pages dans l'ordre popularité.

D) Création du graphe

La génération du graphe est assurée par la classe 'Graph'. Cette classe a deux attributs :

-links : une unordered_map avec des clés de type string indiquant les noms des pages, et des valeurs de type unordered_map qui indiquent, pour chaque clé (considérée comme page 'destination'), la liste de toutes les pages qui ont permis d'arriver sur elle accompagnée de leurs nombres de hits associés.

- nodes : une unordered_map avec des clés de type string indiquant le nom du noeud (correspondant à une page) et des valeurs de type int indiquant le numéro de noeud dans le graphe. Cette structure de données supplémentaire a été

rajoutée vers la fin du développement du projet lorsqu'on a eu à faire face à la contrainte de la syntaxe de GraphViz, afin de faire correspondre à chaque page un label unique dans le graphe.

Cette classe est munie de deux méthodes :

- fillNodes : cette méthode balaie la map links pour enregistrer tous les noms de pages présents en réalisant deux parcours : d'abord pour chaque paire la clé indiquant la page destination, ensuite au niveau de chaque paire toutes les pages qui permettent d'arriver sur elle.

- writeGraph : cette méthode prend en paramètre un nom de fichier et écrit selon la syntaxe GraphViz tous les noeuds présents et les liens entre eux.

V. Résultats

Ci-dessus les résultats des tests que l'on a réalisés sur notre application ainsi qu'un exemple de graphe qu'elle génère:

Test ID	Return code validation	Out result	StdErr result	File creation result	Global result
Test1	1	1	1	2	1
Test10	1	1	1	2	1
Test11	1	1	1	2	1
Test12	1	1	1	2	1
Test13	1	1	1	1	1
Test14	1	1	1	1	1
Test15	1	1	1	1	1
Test16	1	1	1	2	1
Test17	1	1	1	1	1
Test18	1	1	1	2	1
Test19	1	1	1	2	1
Test2	1	1	1	2	1
Test20	1	1	1	2	1
Test3	1	1	1	2	1
Test4	1	1	1	2	1
Test5	1	1	1	2	1
Test6	1	1	1	2	1
Test7	1	1	1	2	1
Test8	1	1	1	2	1
Test9	1	1	1	2	1

Figure 2: résultats des tests.

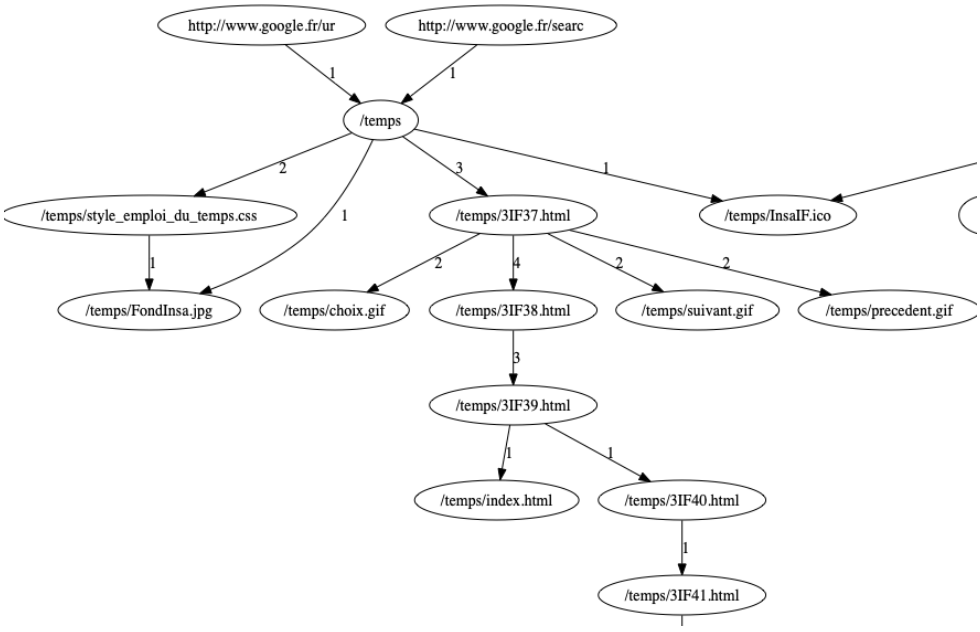

Figure 3: exemple de graphe.

VI. Conclusion et améliorations possibles

En conclusion, le développement de cette application nous a permis de nous familiariser avec la STL et tout ce qu'elle offre, de mettre en place une architecture la plus optimisée possible à l'aide des conteneurs selon le cahier de charges défini.

Evidemment, des améliorations sont possibles. Premièrement, on peut proposer à l'utilisateur plus d'options. On peut par exemple proposer, s'il le souhaite, de choisir le nombre de sites qu'il souhaite afficher dans le classement (au lieu de 10, il peut demander à voir les 15 plus visités par exemple), en ajoutant une option (-n par exemple).

L'avantage du programme étant qu'il est réutilisable, nous entendons par là que le fichier log est entièrement lu et nous avons accès à chaque détail d'un log (IP, referer, GMT, ect …), nous pouvons proposer à l'utilisateur des options pour ajouter plus d'informations au GraphViz ou au classement.

Nous pouvons aussi ajouter une IHM graphique grâce à Qt ou GTK+, qui rendrait l'expérience plus agréable et intuitive (les options seraient par exemple des boutons ou boutons radios).