BuScrapper - dokumentacja

3 lutego 2017

Spis treści

1 Opis programu

Przeznaczeniem programu jest wyodrębnianie rozkładów jazdy ze stron o znanej strukturze, prezentowana wersja jest dostosowana do pobierania krakowskiego mpk, jednak program został napisany w ten sposób, aby możliwie najmniejszym kosztem pozwolić użytkownikowi na pobranie innego typu rozkładów.

Wyodrębnione dane zostają zeskładowane w postaci plików, z których program jest w stanie wyodrębniać informacje dotyczące godzin odjazdów łączących dwa przystanki połączone przynajmniej jedną linią, używając do tego celu wyszukiwarki, które może zostać w łatwy sposób przepisana, aby uwzględniać informacje trudniejsze do wyłuskania.

Program nie posiada interfejsu graficznego, cała komunikacja między użytkownikiem a program odbywa się poprzez plik konfiguracyjny, którego odpowiednie wypełnienie jest kluczowe przy dobrym działaniu programu. Działający program zapisuje wyciągane z internetu dane do plików, natomiast informacje o wyszukanych połączeniach wypisuje na standardowe wyjście.

2 Opis działania

Sam BuScrapper składa się z wielu komponentów, z czego każdy z nich zakłada że otrzymane przez niego dane mają sens w danym kontekście, po czym przetwarza i podaje dalej lub składuje.

Najpierw program przy pomocy Configurator'a zczytuje wszystkie polecenia użytkownika, wyodrębnia z nich wyrażenia XPath elementów, które mają zostać wyciągnięte, specyfikację połączeń, które mają zostać wyszukane i dane potrzebne do konstruowania zapytań dla hosta. Configurator współdziałą z RequestCreator'em w celu stworzenia kolejki zapytań po stworzeniu kolejki obiektów Task, z każdego obiektu task jest tworzona kolejka zapytań wyciągających dane dla danej linii.

Kolejka zapytań jest wykorzystywana przez obiekty DownloadThred, które współdziałają z obiektami SnatchThread na synchronizowanej tablicy implementującej PagesBuffer. Trafiają do niej strony pobrane przez DownloadThread, z których SnatchThread ma wyodrębnić informacje, wyspecyfikowane jako wyrażenia XPath. Następnie SnatchThread uruchamia podaną mu implementację StoreBusInfo, która np. przy podaniu FileStoreBusInfo powoduje zapisanie w katalogu o nazwie numeru linii oraz kierunku, plik o nazwie równoważnej nazwie przystanku i zapisuje w nim numery linii w formacie (bez odstępów) godzina: minutydzieńpowszedni:minutysoboty: minutyniedziele:, jeżeli którejś informacji brakuje w danej godzinie, występują tam znaki ::. FileStoreBusInfo zabezpiecza się przed zeskładowaniem danych niepełnych lub wadliwych.

Gdy wszelkie aktualizacje danych zostaną już przeprowadzone, program wykorzystuje przygotowaną przez Configurator kolejkę wyszukiwań i wypisuje na standardowym wyjściu wyniki przepuszczania wyszukań przez klasę Browser.

3 Opis pliku konfiguracyjnego

Do prawidłowego działania programu muszą zostać wprowadzone następujące linijki (konwencja nazwa pola, znak = i wartość

- UPDATE= - wartość TRUE ustala ponowne przeprowadzenie wyłuskiwania informacji z internetu, każda inna wartość wyłączy tę opcję
- XPATHNAZWA= - wyrażenie XPath opisujące gdzie w przeszukiwanym dokumencie może znaleźć się nazwa przystanku
- XPATHNUMER= - wyrażenie XPath opisujące gdzie w przeszukiwanym dokumencie może znaleźć się numer linii
- XPATHDIREC= - wyrażenie XPath opisujące gdzie w przeszukiwanym dokumencie może znaleźć kierunek linii
- XPATHCZASY= - wyrażenie XPath opisujące gdzie w przeszukiwanym dokumencie może znaleźć się linijka z godziną oraz minutami dla niej
- DOWNLOADMETHOD= - GET lub POST, czyli zapytanie protokołu HTTP, które program będzie wykorzystywał
- PAGEURL= - adres strony początkowej
- HOSTNAME= - nazwa hosta dla którego przygotowujemy zapytania
- ZAKRESLINII= - konwencja liniapoczątkowa:liniakońcowa - określa zakres linii, które mają zostać zaktualizowane
- MAXPRZYSTANKOW= - maksymalna liczba przystanków, która może zostać uwzględniona przy tworzeniu zapytań
- MAXKIERUNKOW= - maksymalna liczba kierunków, które mogą zostać uwzględnione przy tworzeniu zapytań

Aby utworzyć zapytanie musimy wprowadzić linijkę w postaci (bez odstępów):
SEARCH=nazwaprzystankupoczatkowego: nazwaprzystankukońcowego: rodzajdnia: początkowagodzinawyszukiwań: początkowaminutawyszukiwań: maksymalnailośćgodzinodstartowej
Rodzaj dnia: 0 - dzień powszedni, 1 - soboty, 2 - niedziele

4 Opis możliwych zmian w kodzie

Oprócz wykorzystywania zastanego kodu, użytkownik może wprowadzić zmiany w implementacji poszczególnych komponentów. Przede wszystkim może określić własną implementację obiektów SnatchThread oraz DownloadThread, aby w inny sposób pozyskiwać zasoby oraz wyciągać z ich informacje, może wprowadzić własną implementację interfejsu StoreBusInfo dla obiektu SnatchThread, aby składować dane w ustalony przez siebie sposób. Zmiana obiektu Configurator może wpłynąć na generowanie zupełnie nowych formatów pliku konfiguracyjnego, pod warunkiem, że będzie zwracał dla pozostałych komponentów te same informacje, które zwraca zastana wersja Configuratora. Zmiana obiektu Browser może rozszerzyć zdolności wyszukiwania zesładowanych danych. Użykownik może rónież wprowadzić swoją wersję implementacji PagesBuffer'a, pamiętając o tym, aby był kontenerem synchronizowanym.