Übungsblatt 9

Ausgabe: 03.01.2012
Abgabe: 13.01.2012

Aufgabe 1 (HTTP-Webcrawler): (20 Punkte)

Ziel dieser Übung ist es, die Grundlagen des HTTP-Protokoll zu erlernen und ein grundlegendes Verständnis für Webcrawler zu erhalten. Dazu soll ein Programm entwickelt werden, das eine vom Benutzer angebende Seite und alle auf ihr verlinkten Seiten (bis in eine beschränkte Tiefe) nach E-Mail-Adressen durchsucht.

Dabei wird abermals das Master-Worker-Pattern zum Einsatz kommen, um die anfallende Arbeit effizient zu verteilen.

a) (7 Punkte) Implementieren sie die Interfaces *HttpUrl* und *HttpConnection* unter Nutzung des HTTP-Protokolls. Die Implementierung der Klasse *HttpUrl* soll dabei einen Konstruktor besitzen, der lediglich eine URL in Form eines Strings benötigt. Nutzen die ausschließlich *TCP-Sockets* zur Implementierung.

b) (4 Punkte) Entwickeln Sie eine Methode, die die in einem HTML-Dokument enthaltenen Hyperlinks (dies können u.a. Links oder Mail-Adressen sein) findet. Die Methode ist mit der URL des zu durchsuchenden Dokuments parametrisiert; als Ergebnis wird eine sortierte Liste der Links geliefert. Nutzen Sie ihre Implementierung aus (a).

c) (8 Punkte) Implementieren Sie nun mit Hilfe von (b) und dem Master-Worker-Pattern einen E-Mail-Adressen-Webcrawler unter Nutzung Ihrer Implementierung der letzten beiden Zettel. Modellieren dazu einen Task, der für eine gegebene Webseite (Task-Argument)

- Alle E-Mail-Adressen auf dieser Seite sucht
- Alle Links auf dieser Seite sucht

und diese als Ergebnis (Task-Result) zurückgibt. Die gefundenen E-Mail-Adressen sollen nun von Master gespeichert werden. Aus den gefunden Links werden neue Task-Argumente generiert und in den Argumenten-Pool eingefügt, um auch diese Seite zu durchsuchen. Die Suche wird gestartet, indem aus einer Start-URL das erste Task-Argument generiert und in den Argument-Pool eingefügt wird.

Damit die Suche terminiert soll es dem Benutzer möglich sein eine maximale Suchtiefe anzugeben, die aktuelle Suchtiefe muss daher sowohl ein Parameter der Task-Argumente als auch der Task-Results sein.

Achtung

- Achten Sie darauf, das Sie eine URL nicht mehrmals durchsuchen
- Achten Sie darauf, die aktuelle Suchtiefe sowohl im Task-Argument, als auch im Task-Result zu speichern
- Ihre Worker müssen daüber informiert werden, wenn der Job beendet ist. Dafür bietet sich die Nutzung von Poison-Objects an. Dabei handelt es sich um ein spezielles Task-Argument, das dem Worker 'vergiftet' - ihm signalisiert, das der Job erledigt ist. Findet ein Worker ein solches Poison-Object im Argumenten-Pool, legt er es dahin zurück, damit auch die anderen Worker terminieren können und beendet sich nach der De-Registierung beim Master selbst. Um die Worker zu terminieren, muss der Master nun also nur noch ein Poision-Object in den Argumenten-Pool legen und warten, bis sich alle Worker de-registiert haben

d) (1 Punkt) Testen Sie das Programm mit der URL http://www.fu-berlin.de/einrichtungen

optional: e) (3 Punkte) Adressen, die auf den 1. Blick unterschiedlich erscheinen, können trotzdem, aufgrund relativer Pfade, identisch sein (zum Beispiel www.test.de und www.test.de/test/../) Implementieren Sie eine Lösung für das Problem, um die Anzahl der mehrfach auftretenden Adressen zu minimieren.