Programmieraufgabe: Implementierung der Funktionalitäten eines Harvesters in Java

29. Mai 2019

1 Beschreibung

1.1 Ausgangslage

Das vorliegende Skript `harvest.sh` liest in mehreren Schritten aus einer oder mehreren METS-Dateien die im Element <Flocat> jeweils im Attribut xlin:href angegebenen URLs ein und lädt die mit den URLs verknüpften Inhalte auf ein lokales Verzeichnis. Das Skript wird dafür eingesetzt, Inhalte von Open Access Repositorien in die Langzeitverfügbarkeitslösung Rosetta zu übernehmen.

1.2 Aufgabe

Die Arbeit des Skripts soll in mehreren Schritten durch ein Java-Programm nachgebildet werden. Die Übungsaufgabe besteht darin, die Kernfunktionalitäten des Skripts in Java zu implementieren. Dabei sollen möglichst existierende Java-Technologien (Libraries) angewandt werden.

Die gewünschten Module sind nach Priorität geordnet:

1. Harvester-Modul: Nimmt einen URL entgegen und lädt die hinter der URL liegende Webressource auf ein lokales Verzeichnis.
2. METS-Extraktions-Modul: Liest aus einer oder mehreren in einem Verzeichnis liegenden METS-Dateien die im Element <Flocat> angegebenen URL aus und leitet sie zur weiteren Verarbeitung an das Harvester Modul.
3. Konfigurationsmodul: Eine oder mehrere Klassen, mit denen das Programm über eine Textdatei beim Laden konfiguriert werden kann. Wichtig sind hier vor allem die Angaben, wo die METS-Dateien zu finden sind und wo die aus dem Netz geladenen Inhalte gespeichert werden sollen.
4. Wünschenswert wäre darüber hinaus eine Log-Datei, in der Fehler beim Harvesten nachgehalten werden können.

Es stehen ein METS-Beispiel und eine vom Skript produzierte url.txt zur Verfügung. Letztere kann z.B. direkt zur Implementierung und zum Testen des Harvestermoduls verwendet werden, da in ihr die URLs in einzelnen Zeilen aufgelistet sind.

Für die Programmierung soll die IDE Eclipse und das Build-Tool Maven verwendet werden. Beides ist bereits auf dem Rechner installiert. Ein rudimentäres Maven-Projekt ist in der Eclipse eingerichtet.

1.2.1 Unteraufgaben

- Erstellen des Programmcodes in Java für die einzelnen Funktionen des Skripts
  - Harvesten von URLs
  - Extraktion von URLs aus METS-Dateien
  - Werkzeug konfigurierbar machen
  - Festhalten der Ergebnisse der Verarbeitung

2 Hilfen

2.1 Hinweise

1. Eine zentrale Funktion des Skriptes ist das Harvesten von URLs und das Ablegen der Inhalte in ein Verzeichnis. Mit dem Schreiben einer Klasse für das Einlesen könnte begonnen werden. Dafür steht eine Version der durch das Skript erzeugten url.txt zur Verfügung.
2. Das Modul zum Harvesten sollte auch Fehler erkennen (Status-Codes) und mit diesen umgehen können.
3. Für das zweite Modul (METS-Extraktion) bietet sich die Nutzung einer existierenden Bibliothek zur Verarbeitung von XML an.
4. Die Klasse Properties ermöglicht einfache Konfigurationen

2.2 Maven

Maven ist ein Build-Tool, dass unterschiedliche Workflows zum Erzeugen von Java-Compilaten bietet und die Abhängigkeiten zu Bibliotheken managed. Die zentrale Steuerung und Konfiguration erfolgt über eine oder mehrere xml-Dateien mit dem Namen pom.xml. Dort werden alle Abhängigkeiten und Konfigurationen aufbewahrt. Mavens Paradigma ist: Convention over Configuration. D.h., solange in der pom.xml nichts anderes steht, geht Maven davon aus, sich an die für Maven definierten Konventionen halten zu sollen (Verzeichnisstrukturen, Verarbeitung der Goals,...).

Mit dem Konsolenaufruf `mvn clean package` können die Java-Sourcen des Projekts compiliert werden. `compile`, `test`, `install` sind einige Maven Goals die nacheinander ausgeführt werden, wenn das Goal `package` aufgerufen wird (vgl. Maven Lifecycle Introduction). Soll ein spezifisches Format für das Deployment eines Java-Projekts erzeugt werden, können spezifische Goals verwendet werden (Beispiel für ein Projekt, das im Tomcat deployed werden soll: `mvn clean test war:war`, pom.xml - DeepZoomService). Das Maven-Plugin für Eclipse bringt einen speziellen Editor für die pom.xml mit und ermöglicht auch das direkte Aufrufen von Maven aus Eclipse.