Dokumentation
Praktikum Software Entwicklung

Dozent: Prof. Dr. Richard Göbel
Beteiligte Studenten:
Christoph Piechula
Christoph Cwelich
Christopher Pahl
Eduard Schneider
Florian Bauer
Sabrina Biersack

24. April 2012

Inhaltsverzeichnis

Teil I

Spezifikation

1 Beschreibung des Programmablaufs

Über einen konfigurierbaren Crawler können HTML-Inhalte von Webseiten bis zu einer bestimmten Tiefe aus dem Netz in ein Archiv geladen werden. Da dies parallelisiert erfolgen soll, müssen die Daten nach dem Herunterladen von temporären Verzeichnissen in das gemeinsame Archivverzeichnis synchronisiert werden. Der aus der URL extrahierte Pfad der HTML-Dateien wird dabei auf das Archiv abgebildet, wobei jede HTML-Datei in einen eigenen Archivordner verschoben wird.

Beim Crawlvorgang werden zusätzlich Metadaten der HTML-Seiten erstellt. Diese werden in einer Datenbank und als XML-Datei im jeweiligen HTML-Ordner gespeichert. Die Datenbank soll dabei wieder aus den XML-Daten rekonstruierbar sein.

Außerdem sollen Filter eingehängt werden können, die bereits in den TMP-Ordnern ungültige Dateien löschen. Über eine Java-Schnittstelle kann anschließend wieder auf die Daten zugegriffen werden. Hierzu müssen sich Clients über eine vorgegebene Schnittstelle beim Textarchiv anmelden. Die Clients werden anschließend bei Änderungen oder neuen Einträgen benachrichtigt und können sich selbstständig über die Schnittstelle einen beliebigen Stand der Daten herunterladen und an Analysetools weitergeben. Dabei sollen auch neue Dateien den Archivordnern hinzugefügt werden können. Ebenso sollen die oben genannten XML-Daten um neue Nodes erweiterbar sein. Zu Vorführzwecken wird ein Testanalysetool erstellt.

2 Datenmodell der Metadaten

Für vereinfachte Such- und Sortieraufgaben wird bereits ein vereinfachtes Datenmodell festgelegt, welches später auf Datenbank und XML-Daten umgesetzt werden soll und zentrale Metadaten über HTML-Seiten erfassen soll. Die XML-Dateien sollen grundsätzlich um neue Tags/Nodes erweitert werden können, die Datenbank darf davon aber nicht betroffen sein.

Name	Datentyp
URL	String
Titel des Dokuments	String
Dateipfad des Archivordners im Archiv	String
Datum der letzten Änderung	TimeStamp oder Integer
Commit Tag der Versionsverwaltung	String

Tabelle 2.1: Metadaten

3 Anforderungen

Im folgenden sind die Anforderungen an die Software spezifiziert. Soweit möglich wurden die Anforderungen schon in einzelne Komponenten und Module gegliedert. Eine grobe Übersicht gibt auch Diagramm **??**

3.1 Crawlermodul

Dieses Modul soll als eigenständiger Prozess laufen und in regelmäßigen Abständen Crawlvorgänge starten und die Daten in das Archiv schreiben.

3.1.1 Steuerung

3.1.1.1 Config-Datei

Die Steuerung des bzw. der Crawler erfolgt über eine Config-Datei. Darin werden bereits Default- und Fallbackwerte festgelegt Es können folgende Parameter eingestellt werden:

1. Tiefe bis zu der Links gefolgt werden soll
2. Zeitintervalle der Crawlvorgänge
3. maximale Anzahl der gleichzeitig gestarteten Crawlerinstanzen
4. Filtereinstellungen

3.1.1.2 Kommandozeileninterface

Optional können beim Start mittels Kommandozeile zusätzliche Parameter übergeben werden:

- Damit lassen sich Werte aus der Config-Datei überschreiben.
- Es kann eine Liste von Domains übergeben werden, die als Startpunkte für die Crawler verwendet werden sollen. Wobei mindestens ein Element obligatorisch ist.
- Es kann ein Datenbank-Recovery erzwungen werden. siehe **??**

3.1.2 Ausführung des Crawlvorgangs

Die Folgenden Teilvorgänge sind so ausgelegt, dass sie parallelisiert abgearbeitet werden können.

3.1.2.1 Instanziierung

Pro Domain wird eine Crawlerinstanz gestartet bis die Obergrenze an Instanzen erreicht wird. Für die Crawlerinstanzen wird ein externes Tool verwendet (wget). Jede gestartete Instanz kopiert den Inhalt der Seite in je ein temporäres Verzeichnis. Dabei wird die online vorhandene URL-Pfadstruktur der HTML-Dateien auf das Dateisystem abgebildet. Je Domain wird dadurch ein Hauptverzeichnis erzeugt.

3.1.2.2 Bereinigung und Normalisierung

Nun werden die temp-Ordner bereinigt (u.a. leere Ordner entfernt) und die HTML-Dateien in ein Archiv-Ordner gleichen Namens (inklusive Dateiendung) kopiert.

3.1.2.3 Filterung

Die Daten werden in diesem Teilvorgang von Filtern überprüft und ggf. gleich aussortiert.

3.1.2.4 Extraktion der Metadaten

In diesem Teilvorgang werden die Metadateien im XML-Format extrahiert und jeweils als Datei im zugehörigen Archivordner gespeichert.

3.1.2.5 Synchronisation

Zuletzt werden die so vorbereiteten temporären Ordner in das vorhandene Archiv synchronisiert (mit rsync). Dabei wird jeder Domainordner über ein Dateimutex gesperrt, um gleichzeitiges Schreiben zu verhindern. Veraltete Archivordner werden dabei komplett überschrieben (können aber ggf. durch die Versionierung wieder hergestellt werden), sodass auch veraltete Analysedaten gelöscht werden.
Zum Abschluss werden die Änderungen der Domainordner mit einem Commit versioniert.

3.1.2.6 Datenbankaktualisierung

Während oder nach dem Synchronisationsvorgang wird ein Batch von SQL-Statements für die neuen oder geänderten Daten zur Aktualisierung der Datenbank erstellt.

3.1.3 Filter

Filter können über die Konfiguration beim Starten bekannt gemacht werden und werden als Module/Plugins hinzugefügt. Während des Teilvorgangs **??** erhalten diese eine HTML-Datei und geben nach Prüfung einen Wahrheitswert zurück ob die übergebene HTML-Datei behalten werden soll.

3.1.3.1 Testfilter: Werbefilter

Als Testfilter wird ein Filter implementiert, der über eine Blacklist bestimmte Werbedomains aussortiert.

3.2 Archiv

3.2.1 Aufteilung

Das Verzeichnis ist in einzelne Domainordner getrennt.

3.2.2 Versionierung

Jeder Domainordner wird über eine dezentrale Versionsverwaltung (git) versioniert. Damit ist das Wiederherstellen älterer Versionen grundsätzlich möglich, wobei diese aber manuell über die git-Schnittstellen abgerufen werden müssen. Änderungen an Dateien müssen immer mit einem Commit bestätigt werden.

3.2.3 Synchronisation

Bei Schreibvorgängen muss ähnlich wie unter **??** beschrieben, die Daten gegen konkurrierende Dateizugriffe gesichert werden. Dabei wird immer der ganze Domainordner gesperrt.

3.2.4 Dateisystem

Beim darunterliegenden Dateisystem wird von einem vorhandenen Unix-FS ausgegangen.

3.2.5 Komprimierung

Eine explizite Dateikomprimierung wird erstmal nicht vorgesehen, ist aber zum Teil schon durch die Versionierung gegeben, da alte Revisionen gepackt abgelegt werden.

3.3 Programmierschnittstelle - Java-Client/Server

Diese Schnittstelle soll die Anbindung der Analysemethoden ermöglichen und macht gleichzeitig einen Zugriff über das Netzwerk möglich.

3.3.1 Schnittstelle - Client

3.3.1.1 Registrierung am Server

Der Java-Client muss sich am Server mit seiner Netzwerkadresse anmelden, um bei Benachrichtigungen kontaktierbar zu sein.

3.3.1.2 Benachrichtigungen

Sobald serverseitig Änderungen der archivierten Daten vorliegen, müssen alle Clienten das aktualisierte Datum und einen Commit-Tag empfangen können. Mithilfe dieser Daten entscheiden die Clienten, ob die aktuellen Änderungen für das an den Client angebundene Modul (z.B. Analysetool) relevant sind und erzeugen gegebenenfalls SQL-Statements für die Datenbankabfrage.

3.3.1.3 Datenbankabfragen

Über diesen Client können vorbereitete SQL-Statements an einen Java-Server geschickt werden. Die SQL-Abfrage wird soweit vorbereitet, dass nur noch ein SQL-Bedingungsausdruck für die WHERE-Klausel angegeben werden muss. Optional soll auch eine ORDER-BY-Klausel im selben Stil angegeben werden können. Als return-Wert wird eine Liste von Metadatenobjekten zurückgegeben.

3.3.1.4 Dateiabfrage

Mittels der Metadatenobjekte kann man sich über eine gesonderte Anfrage die vorhandenen Archivordner aus dem Archiv nachladen.

3.3.2 Metadatenklasse

Die Metadatenklasse dient nicht nur als Schlüssel zur Dateianfrage ans Archiv sondern auch zur Erweiterung und Auslesen der XML-Daten.

3.3.2.1 Auslesen von Zusätzlichen Tags

Durch Übergabe eines Tagnamens an eine get-Methode wird ein passender XML-Node herausgesucht wird.

3.3.2.2 Erweiterung der XML-Dateien

Mittels einer set-Methode, die Namen und Inhalt des Tags als Parameter erhält, können neue Tags hinzugefügt werden. Die XML-Datei muss daraufhin automatisch gesichert werden.

3.3.3 Server

Im Hintergrund nimmt ein Java-Server die Nachrichten der Clients entgegen und führt die o.g. Funktionen aus und gibt die Ergebnisse oder Fehlermeldungen an den Client zurück. Datei- und Datenbankfunktionen können evtl. mit dem Crawlermodul geteilt werden.

3.3.3.1 Clienten registrieren

Der Server hält eine Liste von angemeldeten Java-Clienten und verwaltet die Verbindungen der Clienten und verwirft sie bei Verbindungsverlust.

3.3.3.2 Update-Notifier

Der Update-Notifier ist ein eigens laufender Thread des Servers, der in einem vorgegebenen Intervall (z.B. stündlich) in der Datenbank prüft ob neue Commit-Tags vorhanden sind. Hierfür speichert er sich den Zeitpunkt der letzten Update-Suche und vergleicht ihn mit dem Datum der Datensätze in der Datenbank. Wird eine Änderung gefunden, werden das Datum und der Commit-Tag zwischengepuffert.

3.3.3.3 Clienten informieren

Sobald die Update-Suche fertig ist, werden die gepufferten Informationen an die regisitrierten Clienten geschickt.

3.3.3.4 Dateien liefern

Von den Clienten können nun Anfragen in Form von vordefinierten SQL-Statements empfangen werden, worauf der Server die zu den Statements gehörenden Dateien den Clients zur Verfügung stellt.

3.4 Test Analysetool

Zur Demonstrations- und Testzwecken der Java-Clientschnittstelle wird ein Analysetool erstellt, welches bestimmte Wörter zählt und das Ergebnis im Archiv als Datei sowie im XML als zusätzliches Tag speichert.

3.5 Datenbank

Die Datenbank dient zur Speicherung der grundlegenden Metadaten und soll schnelle Suchanfragen ermöglichen.

3.5.1 Zugriff

Der Zugriff erfolgt grundsätzlich nur über die bereitgestellten Schnittstellen.

3.5.2 Aktualisierung

Die Datenbank muss beim Fertigstellen des Crawlvorgangs auf den neuesten Stand gebracht werden.

3.5.3 Wiederherstellung

Sollte die Datenbank beschädigt oder geändert werden, dann soll diese wieder aus den XML-Metadaten rekonstruiert werden können. Aus diesem Grund wird erstmal von einer einfachen SQLite-Datenbank ausgegangen, da diese dateibasiert ist und daher relativ simpel erstellt und wieder gelöscht werden kann.

3.6 XML-Dateien

Für die Validierung der XML-Daten muss eine XML-Schema ausgearbeitet werden. Besonderes Augenmerk ist hierbei auf die Erweiterbarkeit zu legen.

4 Entwicklungsumgebung

4.1 Programmiersprachen

Es werden die Sprachen Python und Java benutzt.

4.1.1 Python

Python in der Version 2 (2.7) wird für die systemnahen Teile verwendet, wie das gesamte Crawlermodul, der Zugriff auf das Dateisystem (Archiv) und die notwendige Ordnersynchronisation, da hier bereits leistungsfähige Libraries für den Zugriff auf das Dateisystem, SQLite und die Versionsverwaltung vorhanden sind.

4.1.2 Java

Die Client-Server Architektur der oben genannten Programmierschnittstelle werden mit Java 1.7 umgesetzt. Hierzu ist Java besonders geeignet und es ist gewährleistet, eine an der Hochschule Hof allgemein verständliche Schnittstelle zu schaffen.

4.2 Dokumentation

Die Dokumentation wird in Latex als ein fortlaufendes Gesamtdokument erstellt, welches je nach Phase um weitere Teile erweitert wird.

4.3 Teamsynchronisation

Dokumente und Quellcode werden über ein gemeinsames Repository auf github.com synchronisiert.

4.4 Sprache

Die Sprache der Dokumentation ist Deutsch wobei natürlich geläufige Fremdwörter enthalten sind. Quellcode, Kommentare und daraus abgeleitete APIs (Javadoc und Sphinx) sind in Englisch zu verfassen.

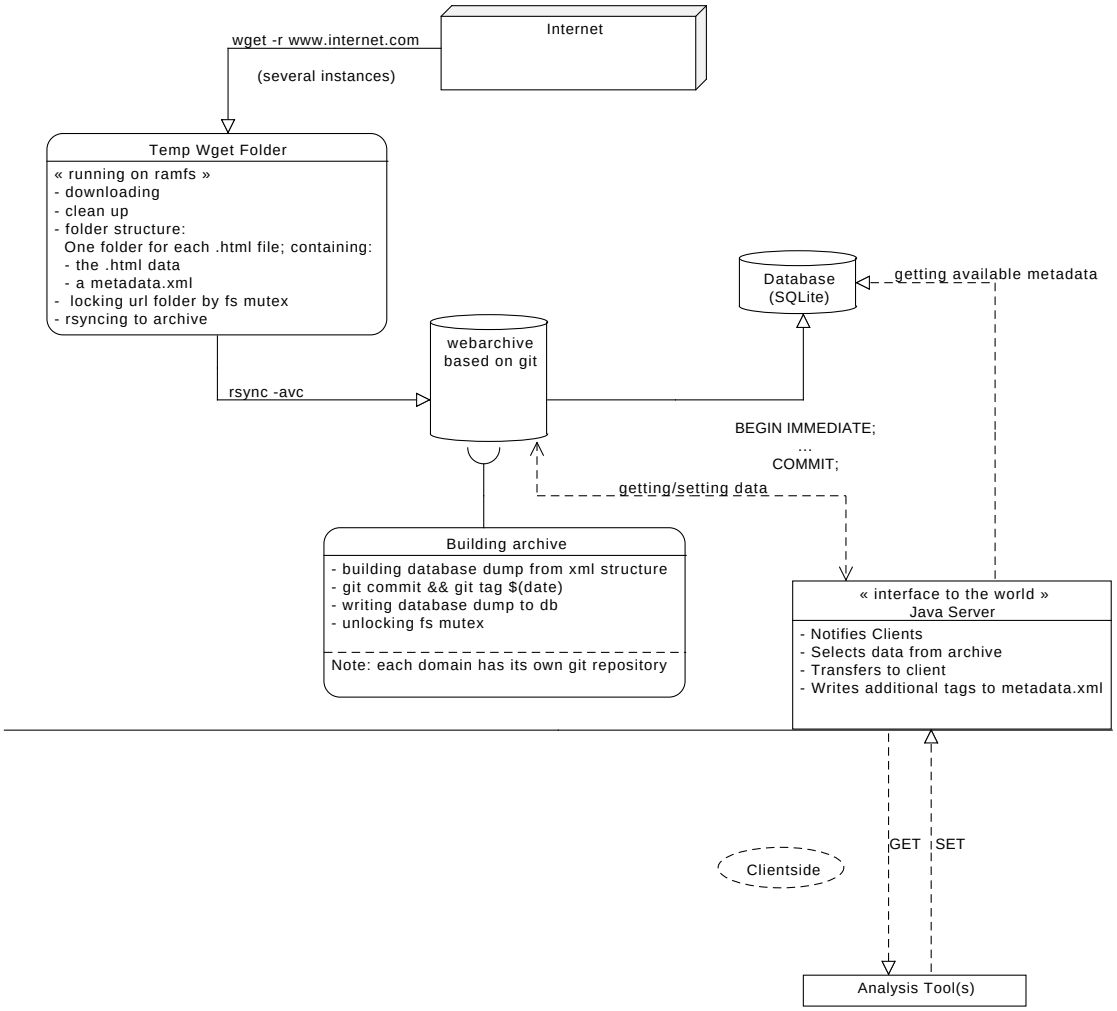

Abbildung 4.1: Diagramm: Grundlegendes Design