

Programmierung 2

Programmieraufgabe "Wordcloud" (vorläufiger Stand)

Prof. Dr. Oliver Hummel





Zum Thema: Ich bin kein Biologe...;-)





Wir wollen ein Tool entwickeln, mit dem klickbare HTML-Wordclouds gebaut werden können

Word- oder Tag-Clouds analysieren eine Menge von Text und stellen die darin enthaltenen
 Wörter je nach Frequenz in verschiedenen Größen dar

add adding at animation center Cloud code color community content cool create CSS data-attribute data-show-value data-weight demo dev development different display focus font-size functionality hrefhtmlimage instead item jan javascript lines links list looks popular profile read rem remove Size style tag terms ul value web weight work

Quelle: https://tagcrowd.com



Funktionale Anforderungen

User Story: Als Professor möchte ich Wordclouds aus meinen Dateien (Folien) erstellen können, um Studierenden einen Überblick über die enthaltenen Themen geben zu können.

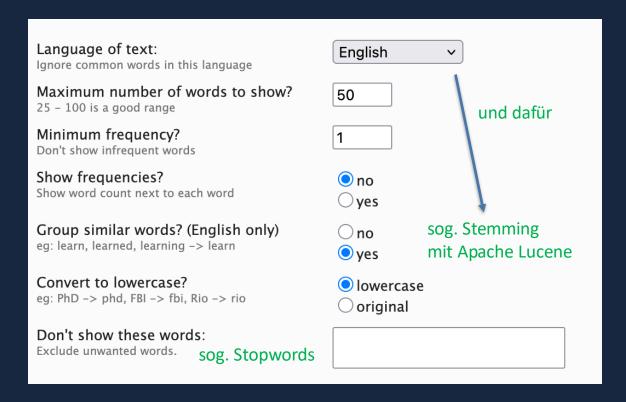
Main Features

- Beliebige Ordner mit folgenden Dateitypen sollen analysierbar sein
 - .txt
 - .pdf (z.B. PDFBox)
 - .docx, .pptx (z.B. Apache POI)
- Link-Vorgabe soll gesetzt werden können, bspw.
 - bspw.: https://www.google.com/search?q=XXXXXXX
- Stopwords sollen über eine Datei gesetzt und über die UI ergänzt werden können
- Generierte Datei im Browser anzeigen: Runtime.getRuntime().exec(command);



Optionen

In Anlehnung an Tagcrowd.com



- Worte alphabetisch sortiert oder nicht
- Statistische Angaben loggen, wie Gesamtzahl der Worte, versch. Worte o.ä.
- (neu): Ausgabe Wordcloud als CSV-Datei, optional als Excel



Weitere wichtige Vorgaben



- individuell zu erstellen und zu testieren
- Verwendung von Branches ist obligatorisch

Nutzen Sie eine Layered Architecture

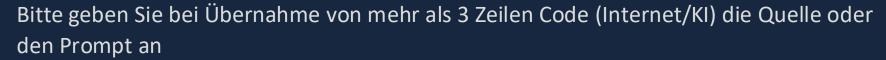
inkl. voll qualifizierten Package-Namen

Erstellen Sie sinnvolle JUnit-Tests für Ihre Domänenlogik

Code-Qualität (PMD) beachten

Optional: Verwendung einer GUI





• z.B. in einer Datei quellen.txt im Ihrem Projektverzeichnis

Verwenden Sie für die Textextraktion bitte Bibliotheken

- Apache POI für Office-Formate
- Apache PDFBox für PDFs
- Apache Lucene f
 ür das sogenannte Tokenizing und Stemming

Für Log-Ausgaben einen Logger verwenden

s. Log4J im MVN-Projekt

Sie erhalten eine vorbereitete HTML-Datei

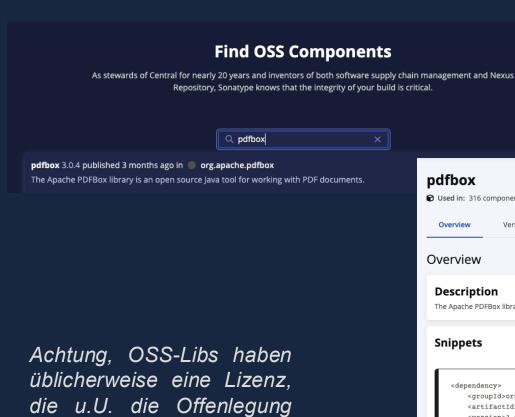
• in diese müssen "nur noch" die Worte mit ihrer Frequenz geschrieben werden



Dependencies finden



Go to: https://central.sonatype.com



des eigenen Quellcodes

notwendig machen kann!

pdfbox 3.0.4 Used in: 316 components Overview Versions Dependents Dependencies Overview Description The Apache PDFBox library is an open source Java tool for working with PDF documents. Snippets Apache Maven Copy to clipboard <dependency> <groupId>org.apache.pdfbox</groupId> <artifactId>pdfbox</artifactId> <version>3.0.4 </dependency>



Minimal-Beispiel Text-Extraktion PDFBox

```
String pdfFilePath = "src/main/resources/some.pdf";
File pdfFile = new File(pdfFilePath);
try (PDDocument document = Loader.loadPDF(pdfFile)) {
 PDFTextStripper pdfStripper = new PDFTextStripper();
 String text = pdfStripper.getText(document);
 System. out. println(text);
} catch (Exception e) {
 e.printStackTrace();
```



Minimal-Beispiel Lucene

```
String text = "Der schnelle braune Fuchs springt über den lahmen Hund.";
try (Analyzer analyzer = new GermanAnalyzer()) {
 TokenStream tokenStream = analyzer.tokenStream(null, text);
 CharTermAttribute termAttribute =
      tokenStream.addAttribute(CharTermAttribute.class);
 tokenStream.reset();
 while (tokenStream.incrementToken()) {
   System.out.println(termAttribute.toString());
} catch (IOException e) {
 e.printStackTrace();
```