Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

Crawling the Web

Felix Elliger & Markus Steiner

Übersicht

- HTML Parser
  - Anforderungsanalyse
  - Probleme
  - Externe Open Source Bibliotheken
  - Jericho HTML Parser
  - Vorstellung unserer Bibliothek
- Wikipedia
  - Möglichkeiten um an die Personendaten zu kommen
  - Auswahl der Daten für unsere Wikipedia-Tabelle

Anforderungsanalyse für den Parser

- **Extraktion von Texten aus dem HTML-String**
  - Body-Text
  - Titel-Text
- **Umwandlung aller HTML-Sonderzeichen in Unicodezeichen**
  - Zum Beispiel *&auml* → ä
  - Ersetzung aller * * durch gewöhnliche Leerzeichen
- **Zuverlässige Ergebnisse auch bei schlechtem HTML Code**
- **spezielle Hervorhebungen im Body oder Titel erhalten**
  - Fett geschriebenen Text
  - Kursiv geschriebenen Text

Probleme beim Parsen

- **Sehr schlechter und nicht standardkonformer HTML-Code**
  - Endtags fehlen z.B. *<body>foobar*
  - Feine Unterschiede z.B. *<body >foobar</body >*

  ➔ reguläre Ausdrücke könnten sehr komplex werden

- **Umwandlung der Sonderzeichen**
  - Erster Ansatz:
    manuelle Definition und Umwandlung von Zeichen wie ÄÖÜ

    aber: was passiert mit **Mario Gómez García** ?

    Folge: Liste der zu definierenden Sonderzeichen wird sehr lang

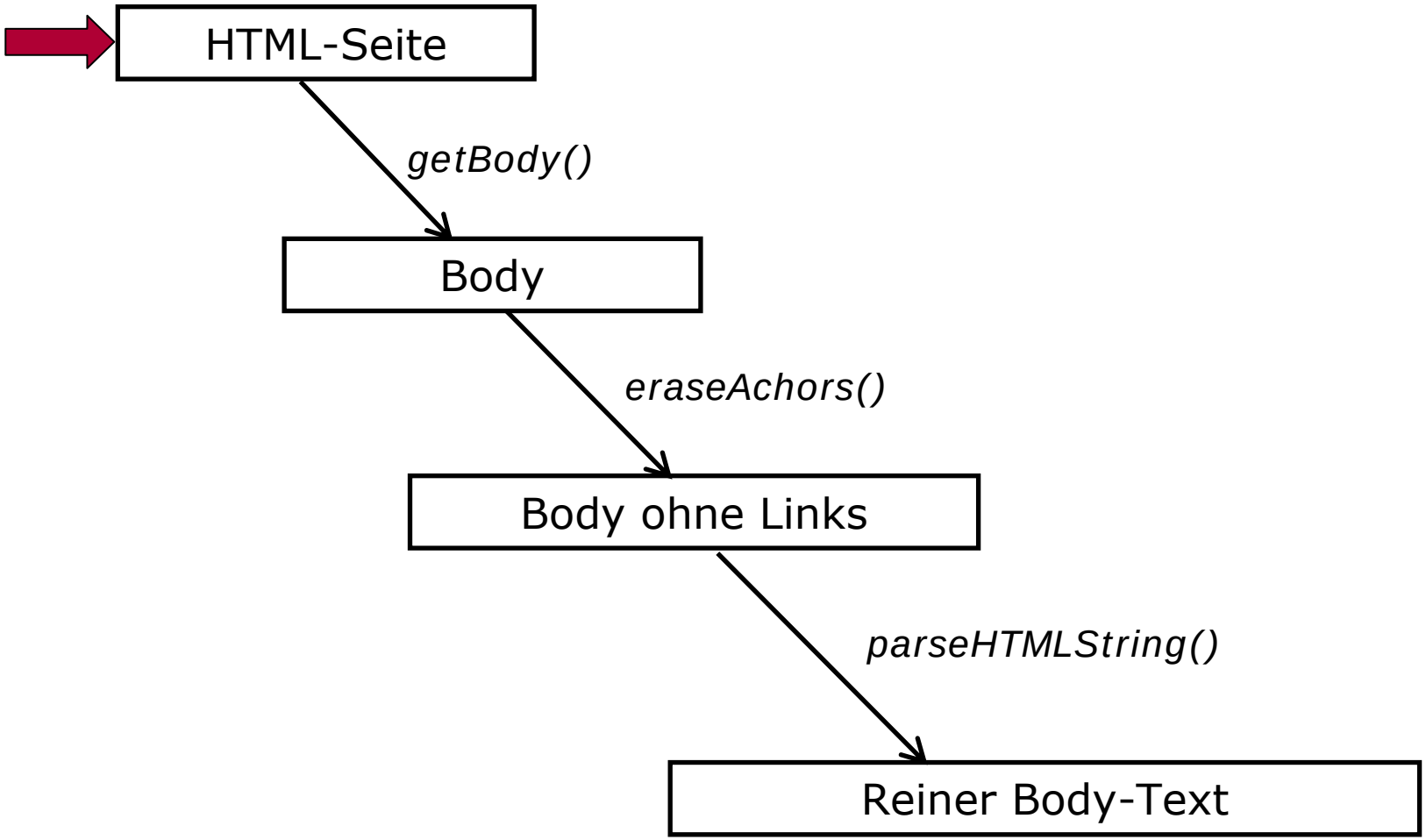
Externe HTML Parser Bibliotheken

- **Beispiele**
  - JTidy, NekoHTML, Java HTML Parser, TagSoup
  - Weitere unter http://java-source.net/open-source/html-parsers
- **Vorteile**
  - Alle Bibliotheken sind Open Source
  - Automatische Korrektur von schlechtem HTML-Code
  - Definitionen durch RegEx entfallen fast vollständig
  - Wesentlich robuster als unsere Definitionen
- **Nachteile**
  - Teilweise sehr schlechte Dokumentation

Jericho HTML Parser 2.5

- Open Source Bibliothek unter GNU Lesser General Public License
- HTML-Dokumente parsen, analysieren und manipulieren
- **Vorteile**
  - sehr robust, d.h. auch bei schlechtem HTML-Code gute Ergebnisse
  - keine riesigen Definitionen von Sonderzeichen
  - gute Dokumentation unter
    http://jerichohtml.sourceforge.net
    http://jerichohtml.sourceforge.net/doc/api/index.html
- **Nachteil**
  - INFO Ausgaben bei schlechtem HTML-Code
    ➔ Fehlerausgabe in Datei

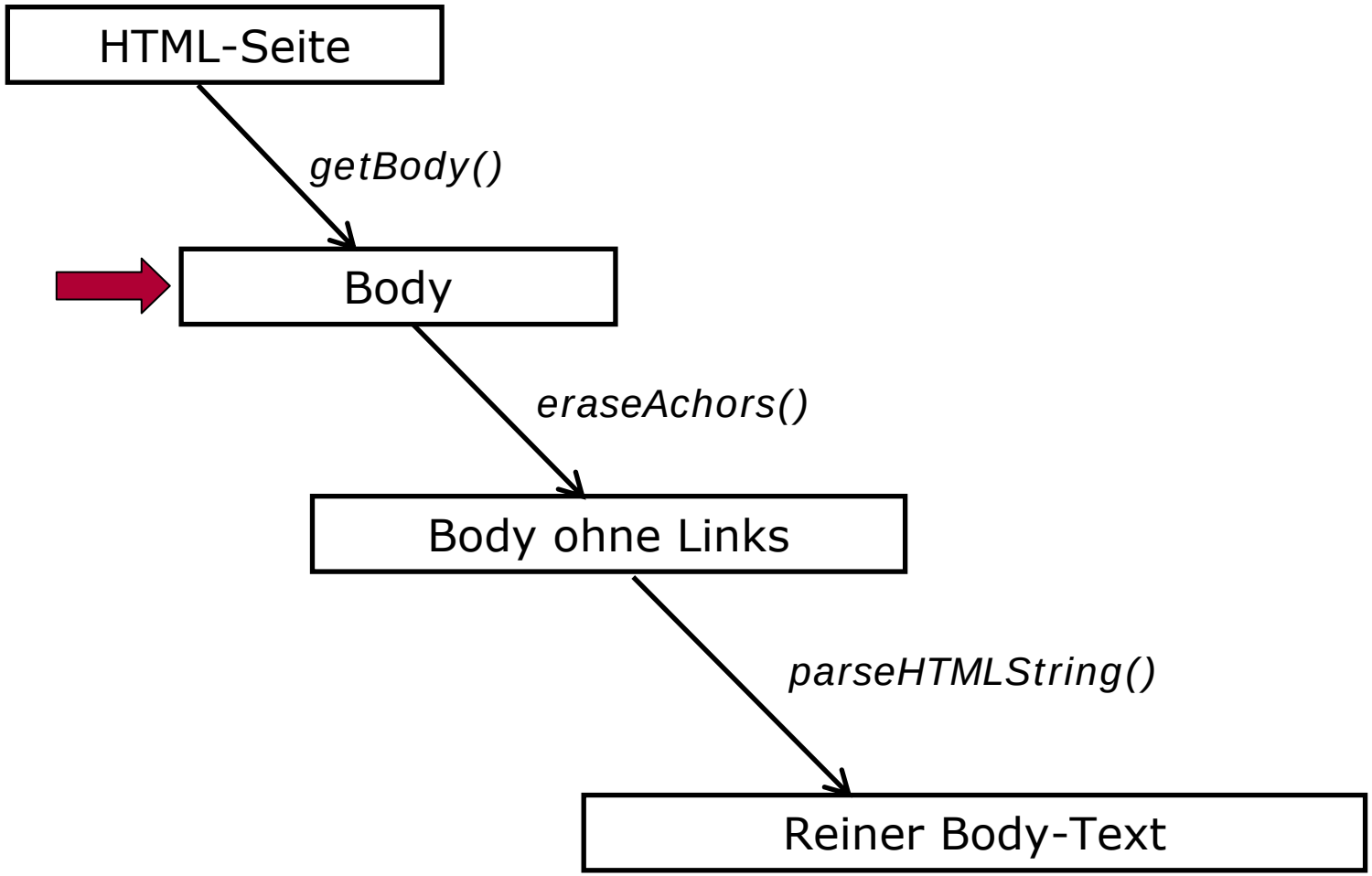
Textextraktion aus dem Body



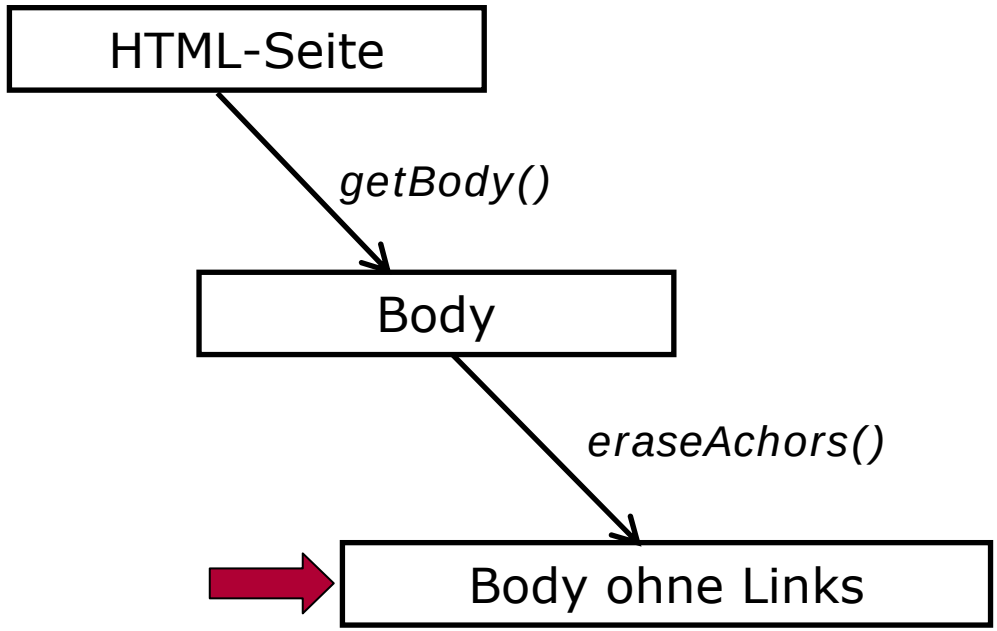
getBody()


```
public static String getBody(String htmlPage) throws TagNotFoundException{
   String s = null;
   Source src = new Source(htmlPage);
   Element bodyElement = src.findNextElement(0,HTMLElementName.BODY);

   if (bodyElement != null) {
        s = CharacterReference.decodeCollapseWhiteSpace
                (bodyElement.getContent());
   }
   else {
        throw new TagNotFoundException("No body-tag found");
   }

   return s;
}
```

Textextraktion aus dem Body

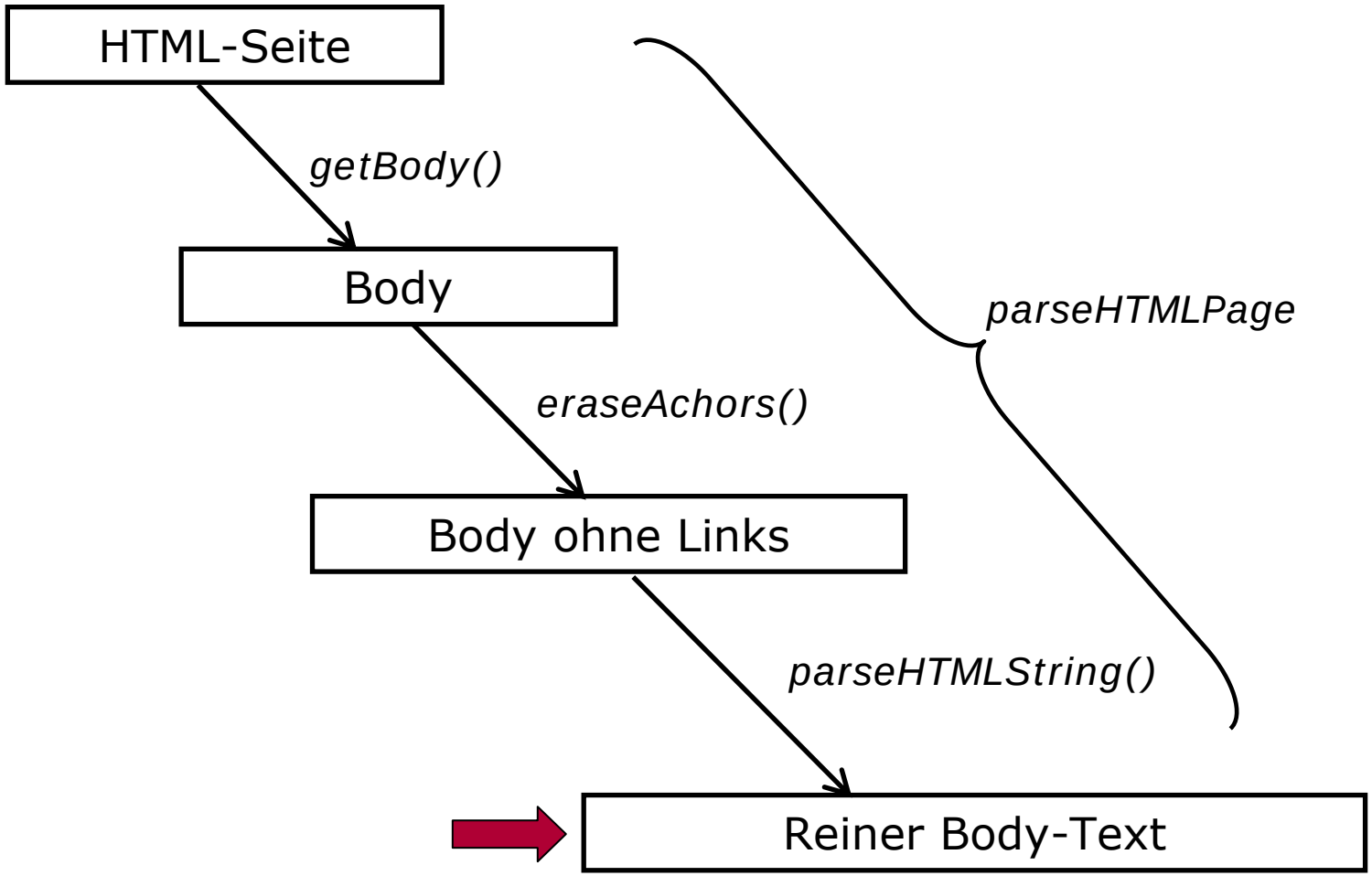


parseHTMLString

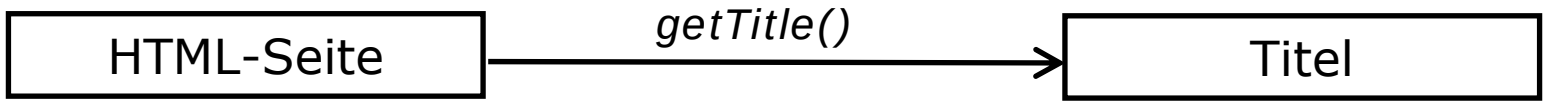



```
public static String parseHTMLString(String htmlString) {
   Source scr = new Source(htmlString);

   return scr.getTextExtractor().toString();
}
```

Textextraktion aus dem Body



Extraktion des Titels – getTitle()




```
public static String getTitle(String htmlPage) throws TagNotFoundException{
    String s = null;
    Source src = new Source(htmlPage);
    Element titleElement = src.findNextElement(0,HTMLElementName.TITLE);

    if (titleElement != null) {
          s = CharacterReference.decodeCollapseWhiteSpace
                  (titleElement.getContent());
    }
    else {
          throw new TagNotFoundException("No title-tag found");
    }

    return s;
}
```

Problem bei getTitle()

- **Quelleninformationen im Titel**
  - Zum Beispiel „ BUNTE.T-Online Newsline: Coroline Beil … “
- **Diskussion: Stören uns diese Informationen später???**
- **Mögliche Lösungen**
  1. **Manuelle Schemata für alle Quellen entwickeln**
     - Sehr aufwändig
     - Nicht robust
       ➔ kleine Änderungen haben große Auswirkungen
       ➔ ständige Wartung nötig
  2. **Vergleichen aller News eines Tages und entfernen gleicher Titelelemente**
     - Automatische Anpassung an Formatänderungen
     - Sehr robust

Diskussion weiterer Methoden

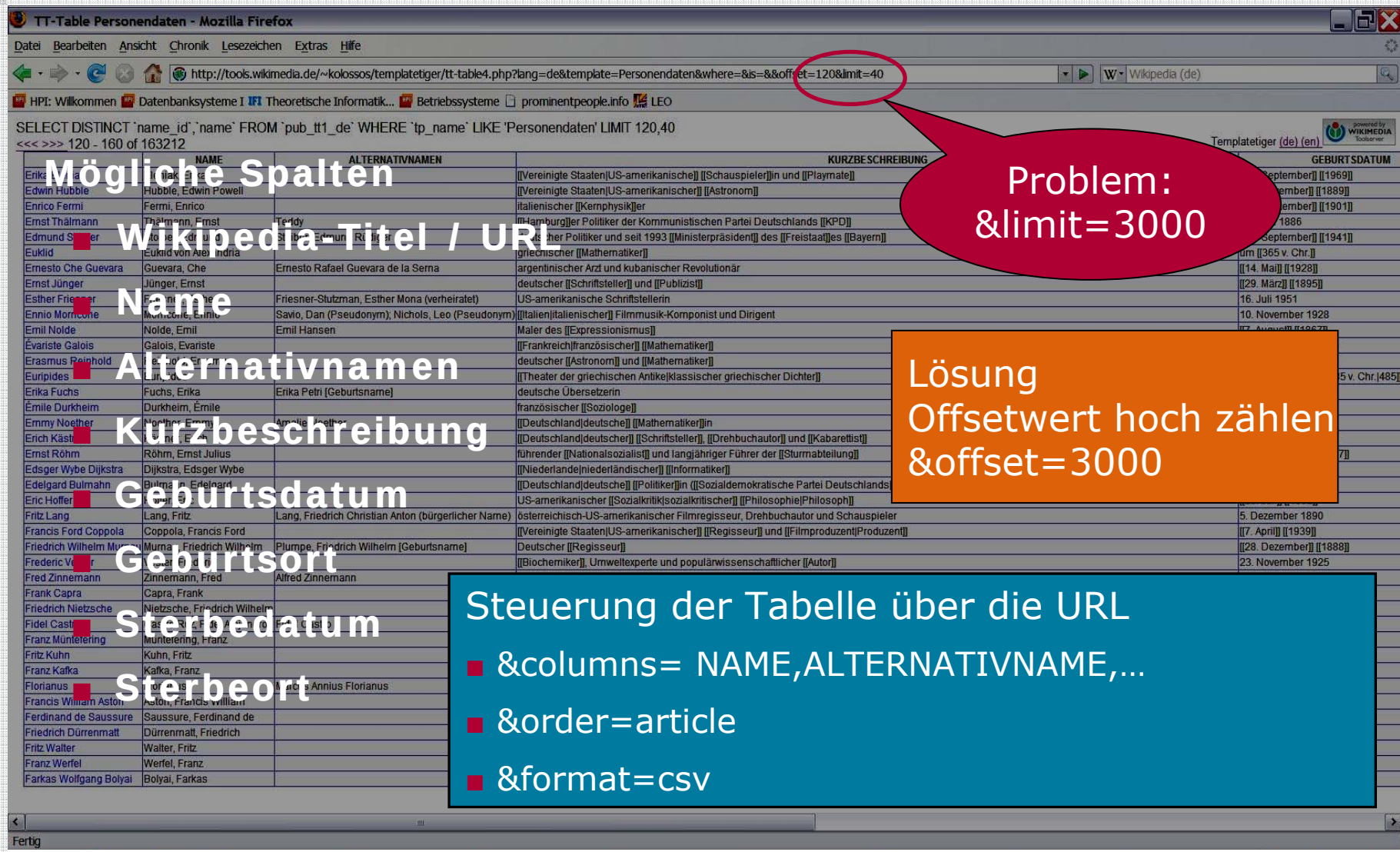
- **Tokenize**
  - zerlegt den String in ein Array einzelner Worte
  - Trennung erfolgt an Leerzeichen
- **ModifyPunctuation**
  - Entfernt Satzzeichen
  - Ersetzt ` durch ‘‘
- **getUpperCaseWords**
  - Liefert alle großgeschriebenen Worte
- **getAllTextsIn**
  - Liefert alle Textelemente zwischen bestimmten Tags

Wikipedia

3 Möglichkeiten um an die Daten zu kommen

1. **Tabelle**
2. **CSV**
3. **Direkter Zugriff auf Toolserver**

Parsen der Tabelle



Parsen der CSV-Datei

Probleme

- Schlechte Qualität
  ➔ Keine Kommas
- Limit bei 3000
  ➔ auch hier muss der Offsetwert hoch gezählt werden

Ausgabe:

Andy Warhol, Warhol, Andy Warhola, Andrew , Begründer und bedeutendster Künstler der [[Pop-Art]] , [[6. August]] [[1928]] , [[Pittsburgh]] , [[22. Februar]] [[1987]] , [[New York City|New York]], [[New York (Bundesstaat)|N.Y.]]

Direkter Zugriff auf Toolserver

Wikimedia Toolserver

- Besteht aus aktuellem Datenbank-Dump
- Theoretisch besteht Möglichkeit direkt auf Toolserver zuzugreifen

Vorteil

- Sehr elegante Lösung
- Relativ einfacher und schneller Zugriff auf Personendaten

Nachteil

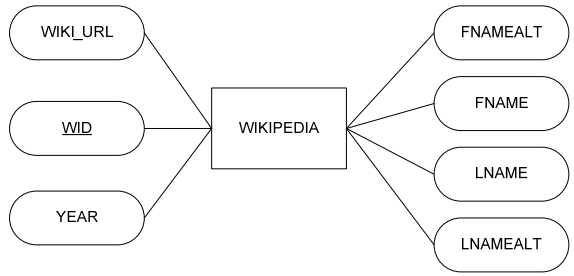
- Bisher nur sporadische Reaktion von Wikimedia-Admin auf die Zugriffsanfrage

Auswahl der Spalten

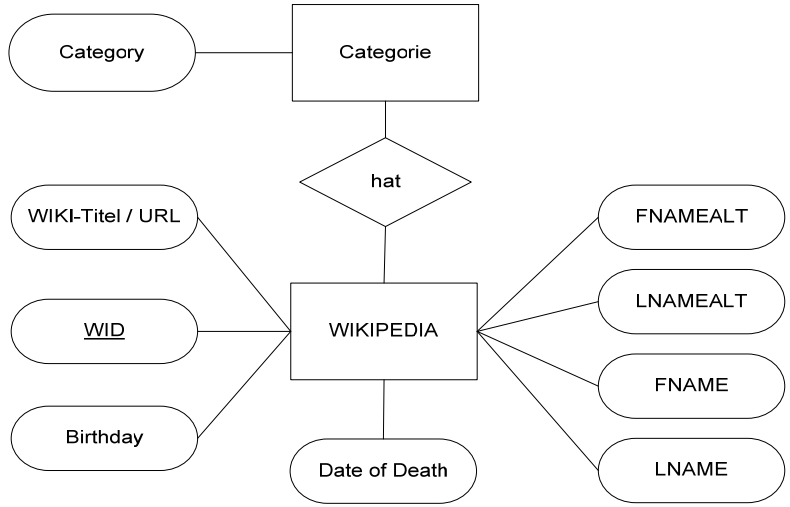
Mögliche Spalten

- Wikipedia-Titel / URL
- Name
- Alternativnamen
- Kurzbeschreibung
- Geburtsdatum
- Geburtsort
- Sterbedatum
- Sterbeort

Unser bisheriger Datenbankentwurf



Unser Vorschlag



Auswahl der Daten

- **Bisher ca. 92.000 Personnamen in unserer Wikipediatabelle**
- **Theoretisch 163.212 Personennamen möglich**
  - **Aber:**
    Brauchen wir
    - Römische Kaiser
    - den achten Kalif der Umayyaden usw.
    - **Gegenbeispiel**: Was passiert, wenn Goethe Geburtstag hat
  - Ab welchem Geburtsdatum / Sterbedatum nehmen wir Personen auf?

Quellen

- Mining the Web
- Java ist auch eine Insel
- http://jerichohtml.sourceforge.net
- http://jerichohtml.sourceforge.net/doc/api/index.html
- http://de.wikipedia.org/wiki/Wikipedia:WikiProjekt_Vorlagenauswertung