

Übung Netzbasierte Informationssysteme

Termin 6: Web Suche

Prof. Dr. Adrian Paschke

Arbeitsgruppe Corporate Semantic Web (AG-CSW)
Institut für Informatik, Freie Universität Berlin

paschke@inf.fu-berlin.de

<http://www.inf.fu-berlin.de/groups/ag-csw/>



Agenda

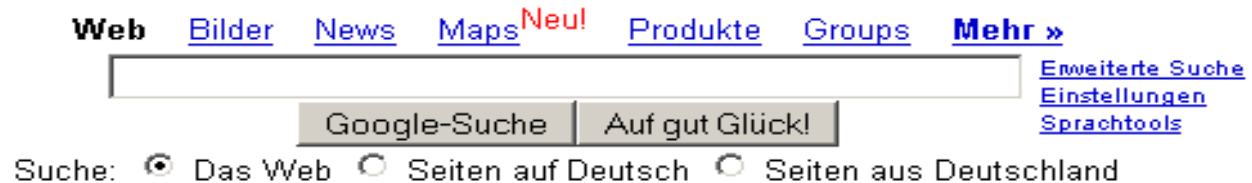
- Feedback Übungsblatt 5
- Thema des heutigen Übungsblatts:
 - Einfache Lucene-basierte Web Suche
 - Hintergrund: Web Search
 - Hintergrund: Search Strategies
 - Hintergrund: Search Queries
- Übungsblatt 6
 - Aufgabe: Lucene
 - Aufgabe: Lucene Indexing
 - Aufgabe: Lucene Querying

Feedback - Übungsblatt 5

- **Lernziele des 5. Übungsblattes waren:**
 - Semantic Web Programmierung
 - Ontologien
 - Regeln
 - SPARQL
- **Wie war`s?**
 - Fehlerquellen
 - Probleme?
- **Abgabe verlängert bis 6.1.2009**

Web Search

When one thinks today of „Web search“, there is really just one word:



„To google“ added to the [Oxford English Dictionary](#) on [June 15, 2006](#) and to the 11th edition of the [Merriam-Webster Collegiate Dictionary](#) in [July 2006](#)

„Googeln“ in die 23. Auflage von Duden aufgenommen

What was there before Google?

[Serious Sports Fans Only \\$1,000,000 in Cash and Prizes!](#)

[For serious sports fans only! Play Fantasy Football!](#)



**It's amazing where
Go Get It will get you.**

Find:

Go Get It

[Enhance your search.](#)



[New Search](#) . [TopNews](#) . [Sites by Subject](#) . [Top 5% Sites](#) . [City Guide](#) . [Pictures & Sounds](#)

[PeopleFind](#) . [Point Review](#) . [Road Maps](#) . [Software](#) . [About Lycos](#) . [Club Lycos](#) . [Help](#)

Importance of Web search

- Search engines are the second most common Web activity after e-mail
- 90% of users find new sites by search
- 80% of Internet traffic to sites comes through search engines

- In August 2007, 95% of Web users performed a search
- A total of 61 billion searches were performed

Importance of relevance

- 80% of users only look in the first two pages of search results (Google: first 20 results)
- Number of matches to typical search terms:
 - wetter : 71 000 000
 - weihnachten : 23 000 000
 - harry potter : 75 600 000

Getting to Google's No 1

- Paid placement: e.g. Google AdWords
- Set up a dummy site with multiple links to our content, multiple text terms and a redirect, etc.: Google gets wise to this
- Hack Google: probably not possible
- Pay a search engine optimiser
- Or just do it yourself
 - Understand how search engines index and answer queries
 - Make use of content, markup and metadata on your site
 - Promote your site to get more links to it from other sites

Search Strategies: Text Indexing

- Text Indexing is the basic approach of all Web search engines
 - Language-specific stemming/stripping
 - Frequency of keywords
 - Location of keywords (closer to the top = more relevant)
- Search engine optimisation
 - Choose your keywords (at least two words)
 - Use text instead of textual graphics
 - Provide static pages, beside dynamic content
 - Scripts and tables can push keywords further down the page

Search Strategies: Markup

- Which markup is relevant to search?
- In HTML, certain elements indicate greater importance of a term
 - <TITLE>
 - <H1>
- In XML, more difficult as the search engine must know how to interpret the XML markup

Search Strategies: Links

- The web is a hypermedia system, i.e. content has links to other content
- Links can be seen as an indicator of importance
 - the more links to some content, the more important that content must be
 - the better the linking site, the better the linked site
- XLinks would provide further information but nobody uses them and no search engine considers them
- Other ways to gain information about the link must be found
 - Standard: the text within the HTML <A> tag describes the content pointed to by that tag
- Google took this principle and built PageRank around it

Search Strategies: Metadata

- What metadata is available in Web content?
- In HTML, one has the <META> tag
 - META name=„description“ content=„.....“
 - META name=„keywords“ content=„.....“
 - Other META tags could be used for internal search functionality e.g. author, date
 - By Oct 2002 only one search engine – Inktomi – crawled the META tag
- There are other approaches which have not yet gained widespread usage in the Web
 - Microformats
 - RDF (e.g. RDFa as RDF in HTML, GRDDL to extract RDF from microformats)

Search Queries

- Problem: Ambiguity of Natural Language
Term „Apache“

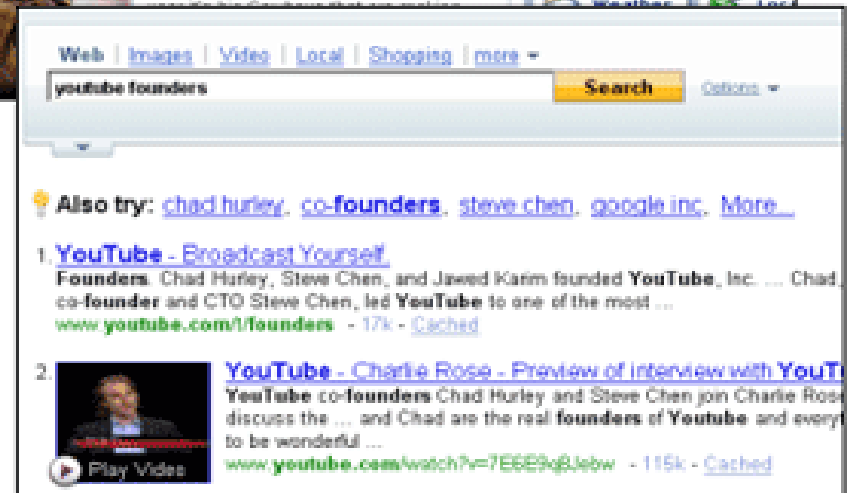
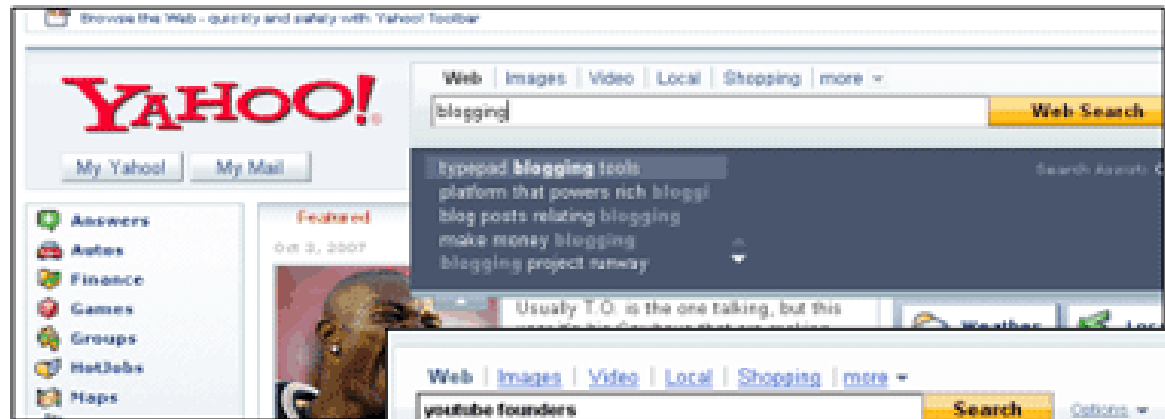


Query Completion

- While the query is still not complete, suggest already suitable completions
- Based on most common query terms from all users

Yahoo offers „Search Assist“ since July 2007 from yahoo.com

„a 61% increase in successful task completion,“ - Yahoo



Clustering

- Gathering of results into groups around a certain theme
- Based on text analysis: which terms occur more often in combination with the query term



The screenshot shows the Clusty search engine interface. At the top, there is a navigation bar with links for 'web', 'news', 'images', 'wikipedia', 'blogs', 'jobs', and 'more >'. The Clusty logo is on the left, and a search box contains the query 'apache'. Below the search bar, there are tabs for 'clusters', 'sources', and 'sites'. The 'clusters' tab is active, showing a list of clusters under the heading 'All Results (208)'. The clusters listed are: Apache Software (22), Apache Indian (15), Framework, Open Source (16), Apache Tribe (9), Perl (7), Apache Nation (7), Filters (7), Reviewed (5), Apache Module (5), and AH-64 Apache (6). There are links for 'more' and 'all clusters'. To the right of the clusters, there is a section titled 'Top 208 results of at least 33,310,000 re'. It lists three search results:

1. [Welcome! - The Apache So](#)
Oil and natural gas exploration and c
newsletter, and news photos.
[www.apache.org](#) - [cache] - Live, As
2. [Apache Web Server Project](#)
The **Apache** HTTP Server Project is
Server ("**Apache**").
[httpd.apache.org](#) - [cache] - Ask, Of
3. [Apache](#)  
 **Apache** (apparently
name given to severa
[America](#), who speak
people. The Apache
Noted leaders have ir
to be fierce warriors and skillful strat
[en.wikipedia.org/wiki/Apache](#) - [cach

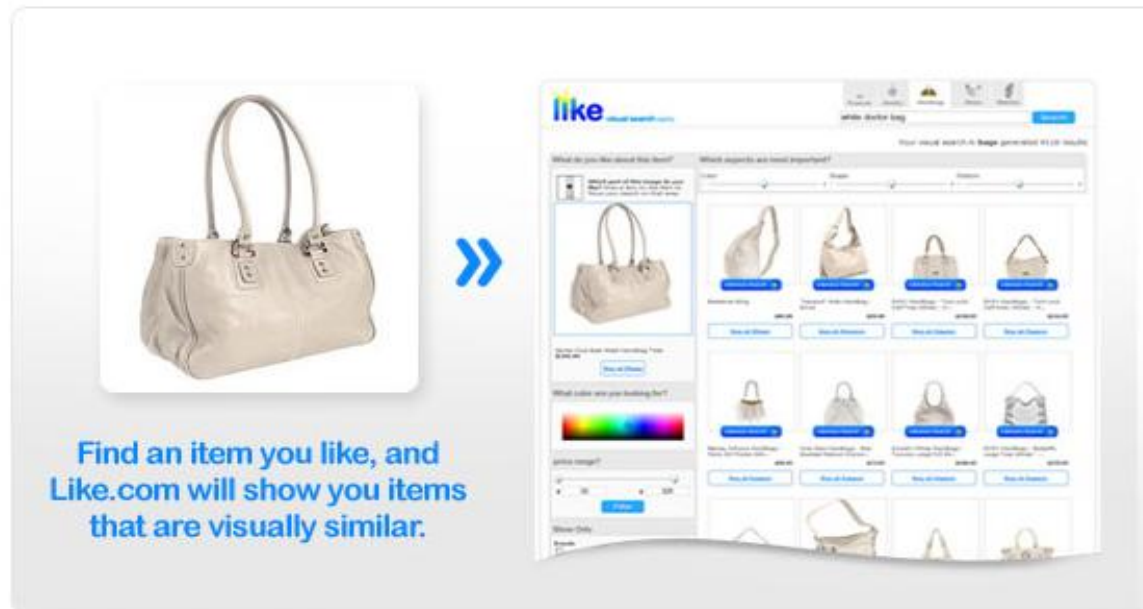
Advanced Search

- Already many advanced searches possible but only if one knows the syntax, for example in Google:
- **Fuzzy Search** - Search for [~music player](#), and google searches for music player, mp3 player, audio player, and other words that have similar meanings to music.
- **Number Ranges** - Searches for a range of numbers, for example [mp3 player 20..60 GB](#) will match 20, 40, and 60GB mp3 players (and any other number inbetween).
- **Wildcard** - When you place a * in your query, google will match any word in between. For example: [apple * player](#) matches apple ipod player, apple mp3 player, etc.
- **Search Page Title only** - Use the `allintitle:` operator to search only page titles. For example: [allintitle:ipod](#) Searches for any page with ipod in the title.

courtesy <http://www.petefreitag.com/item/473.cfm>

Media Search

- Search could be based on feature similarity (like.com)



- However, classical search is still text based
- How to index non-text media for text-based search?

Use text

- Strategy of the major search engines, e.g. Google
„Google analyzes the text on the page adjacent to the image, the image caption and dozens of other factors to determine the image content „ (Google FAQ)

- Identify text **within** the image



SUCHERGEBNIS 1-15 von 28,279



icekingc1.jpg
609 x 1024 - 8602k
joelnylund.com/...
...christmas.html



king_kong_s.jpg
70 x 100 - 21k

www.plakatsammler.de/...
...filmplakate_k.htm



0182da.jpg
749 x 977 - 1462k

www.euroyellowpages.com/...
...grovest0182d.html



KING M.jpg
2592 x 1944 - 1083k



King & King's
green design
initiative
GreenButton.jpg
107 x 144 - 38k



gan_800.jpg
800 x 600 - 799k

Use markup

- **IMG SRC** attribute: use file name as approximate descriptor of the image
- **IMG ALT** attribute: provides short description of the image which may be indexed

Use metadata

- Popular approach but metadata is lacking
 - Often only low level and creation information e.g. audio recording at which KHz, photo taken with which camera
- Audio „signatures“ can identify licensed music
 - Obviously not valid for other types of audio
- Extraction of media metadata ongoing research problem
 - Speech recognition
 - Visual feature extraction
- Other problems
 - Agreeing on a metadata standard for media
 - Agreeing on schemas for the metadata concepts
 - Agreeing on how one binds a media object to its metadata

Metadata: tagging



↑ hochgeladen am 20. Mai 2007
von [davidanthonyporter](#)

Fotostream von davidanthonyporter

452 Fotos

durchsuche

Tags

Dieses Foto gehört

The Apache Trail

durchsuche

Teil von: [Arizona](#)

- The Apache Trail
- apache lake
- phoenix
- scottsdale
- arizona

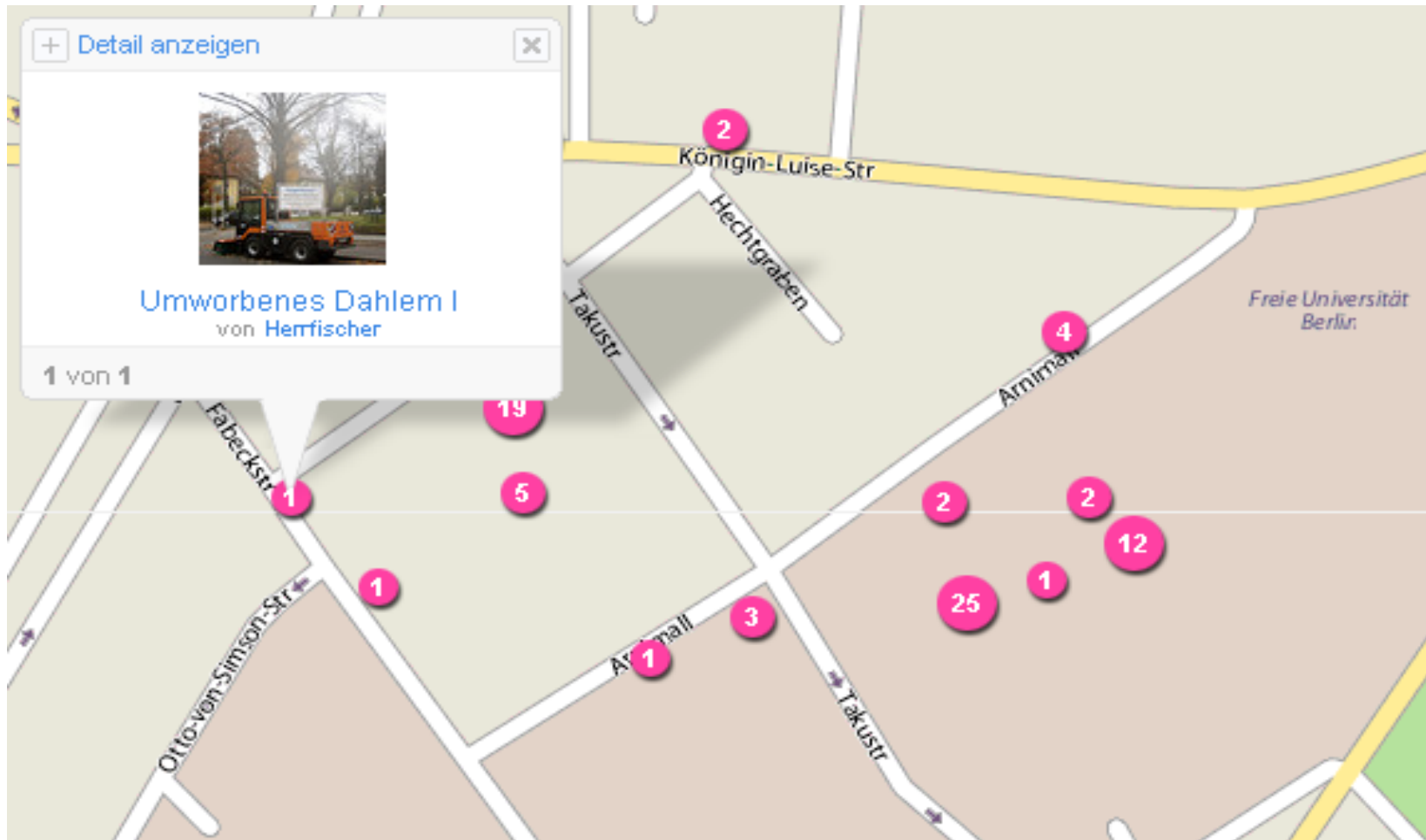
Tags

- The Apache Trail
- apache lake
- phoenix
- scottsdale
- arizona

Metadata: geotagging

- Idea: standardized provision of location metadata
 - Geotagging defines two tags
 - geo:lat=x where x is a real numeral specifying latitude
 - geo:long=y where y is a real numeral specifying longitude
 - These two tags should be attached to media with location property, e.g. photos (Flickr) and videos (YouTube)
 - Enable machines to identify media close to a given location
 - Flickr, give me photos taken close to Times Square, New York
- Supported by geocoding services
 - Map an address to a latitude/longitude pair

Metadata: geotagging (2)



Website Search

- Search restricted to a certain web site
- Search box generally found on the same web site
- Enables a visitor to a web site quicker location of the information desired

Website Search: How To

- Create an index of the website
 - Index the text
 - Use markup
 - Use links
 - Use metadata
- Provide search functionality
 - A HTML search box is just the front end
 - The back end searches for the query string in the index
 - Results are collected and returned
 - The website in which the match was found
 - The snippet of the website which is the match
 - Possibly other relevant (meta)data about the result

Lucene

- A Java-based indexing and search technology developed by the Apache Foundation ... <http://lucene.apache.org/>
- Getting Started
- How Lucene works
 - Text Indexing
 - Querying a Lucene index
- Lucene as web application
- Tasks
 - Customization for the index
 - Customization for the queries

Getting Started

- Lucene 2.4.0 available from <http://www.apache.org/dyn/closer.cgi/lucene/java/>
- Requires Java 1.4, Ant 1.6.5 or up
- Set both JAR files on your CLASSPATH
- Test the installation
 - `java org.apache.lucene.demo.IndexFiles {full-path-to-lucene}/src`
 - `java org.apache.lucene.demo.SearchFiles`

How Lucene works

- Check `src/demo` for sample code to index and search files



html		Folder
DeleteFiles.java	3 KB	JAVA File
FileDocument.java	3 KB	JAVA File
HTMLDocument.java	4 KB	JAVA File
IndexFiles.java	4 KB	JAVA File
IndexHTML.java	6 KB	JAVA File
SearchFiles.java	6 KB	JAVA File

- `IndexFiles` is the main class responsible to create indices.
- `FileDocument` is the indexed object, representing its content as well as creation time and location.
- `SearchFiles` is the main class responsible to search the index and return query results.

Index the data

- Create a new Lucene index using an `IndexWriter`
 - Create a Lucene `Document`
 - Add the document to the index
 - Optimize and close the `IndexWriter`
-
- `StandardAnalyzer` is little more than a Java Tokenizer, converting all strings to lowercase and filtering out useless words and characters (e.g. the, s)
See the `*Analyzer.java` sources under svn:
`contrib/analyzers/src/java/org/apache/lucene/analysis`

Querying a Lucene index

- Open a Lucene `IndexSearcher`
- Construct a Lucene query. An analyzer is used to interpret the query text in the same way the documents are interpreted.
- Perform the search
- Display the top Lucene hits

Lucene query types

- Field query (based on Document field)
 - Title:“This is my website“
- Wildcard query (? – single; * - multiple char wildcard)
 - Test*
- Fuzzy query
 - Roam~
- Proximity query
 - „jakarta apache“~10
- Range query
 - Title:[Aardvark TO Deer]
- Boolean query (AND, NOT, OR, +, -)
 - Apache AND (Jakarta OR Tomcat)
- See [docs/queryparsersyntax.html](https://lucene.apache.org/core/3.6.0/docs/queryparsersyntax.html) for full information

Lucene result ranking

- `hits = searcher.search(query);`
 - `org.apache.lucene.search.Hits` class, it is a collection which can be iterated over
- `hits.score(i);`
 - Each hit has a score, it is a value between 0 and 1
- `searcher.explain(query,i);`
 - Low level explanation of how a document scored for a query e.g.
 $0.29372874 = (\text{MATCH})$
fieldWeight(contents:seoul in 0), product of:
 $1.7320508 = \text{tf}(\text{termFreq}(\text{contents:seoul})=3)$
 $1.3566749 = \text{idf}(\text{docFreq}=6)$
 $0.125 = \text{fieldNorm}(\text{field}=\text{contents}, \text{doc}=0)$
- It can be seen that tf/idf measure is used

Lucene as Web application

- Copy `luceneweb.war` to your Tomcat `webapps` directory
- Create an index from your web site's `webapps` subdirectory

```
java org.apache.lucene.demo.IndexHTML -create -  
index {index-dir} ..
```


- Edit `configuration.jsp` in `webapps/luceneweb`

```
String indexLocation = "/opt/lucene/index";
```

- You may also wish to update the war file by typing `jar -uf luceneweb.war configuration.jsp` from the `luceneweb` subdirectory.

Lucene web app structure

- Check src/jsp for the code for the web app



WEB-INF	Folder
configuration.jsp	1 KB JSP File
footer.jsp	1 KB JSP File
header.jsp	1 KB JSP File
index.jsp	1 KB JSP File
README.txt	1 KB Text Document
results.jsp	9 KB JSP File

Most of the functionality is in results.jsp

- It constructs an `IndexSearcher` with the `indexLocation` that was specified in `configuration.jsp`
- It constructs a query parser only for the document contents
- Results are returned in the collection called `hits`
- An iterator constructs the results table with „known“ fields of the documents such as URL and title

Lucene website search

- Browse to the url `http://localhost:8080/luceneweb`

Welcome to the Lucene Template application. (This is the header)

Search Criteria

Results Per Page

Apache Lucene Template WebApp 1.0

- Try searching for some term in your website!
- Error opening the index? Check `configuration.jsp`

Customization

- We are going to customize the web application so that we can perform better indexing and search over our web sites.
- For example, compare IndexHTML and IndexFiles, and HTMLDocument and FileDocument
- HTMLDocument contents are delivered by HTMLParser

html	Folder
DeleteFiles.java	3 KB JAVA File
FileDocument.java	3 KB JAVA File
HTMLDocument.java	4 KB JAVA File
IndexFiles.java	4 KB JAVA File
IndexHTML.java	6 KB JAVA File
SearchFiles.java	6 KB JAVA File

Entities.java	9 KB	JAVA File
HTMLParser.java	20 KB	JAVA File
HTMLParser.jj	10 KB	JJ File
HTMLParserConstants.java	2 KB	JAVA File
HTMLParserTokenManager.java	44 KB	JAVA File
ParseException.java	7 KB	JAVA File
ParserThread.java	2 KB	JAVA File
SimpleCharStream.java	10 KB	JAVA File
Tags.java	2 KB	JAVA File
Test.java	2 KB	JAVA File
Token.java	3 KB	JAVA File
TokenMgrError.java	5 KB	JAVA File

Customizing Lucene

- Typically you will adapt some code
 - IndexMyWebsite JAVA file
 - MyWebsiteDocument JAVA file
 - SearchMyWebsite JAVA file
- Adapting Lucene to other document types:
 - XML
 - Images
- Extending Lucene for existing document types (HTML):
 - Use the markup
 - Use metadata

Adapting to other document types

- As with HTML, you will probably need to code an IndexMyType Java file and a MyTypeDocument Java file
- With XML, you can use an existing XML Parser. As your XML is very small, it doesn't really matter if it is DOM or SAX.
- Basic idea:
 - Parser „listens“ for your XML content's root element
 - When found, create a Document object with appropriate fields and contents (e.g. the address as one text string, or separate fields for separate values)
 - Add Document to the index
- With Images, you could parse HTML for IMG tags and generate Documents for the index with SRC and ALT values

Extending Lucene for a document type

- HTML results ranking could make use of the markup and/or metadata
- e.g. give more weight to a match in TITLE than in the document body
- How to do this?
 - No need to change the `IndexSearcher` class
 - Easier to „alter“ the rankings after the results have been fetched
- One possibility: separate indexes and merge the results
- Another possibility: use a single index, place markup or metadata that weights the result into a Document field, check for text matches and „boost“ the score accordingly

Übungsblatt 6

- **Lernziele**

 - Einfache Lucene-basierte Web Suche

- **Termine**

 - Ausgabe: 16.12.2008 Abgabe bis: 6.1.2009, 16:00 Uhr
 - 2-wöchiges Übungsblatt
 - 2er-Teams

Viel Spaß und Erfolg !

