# Computer-based Content Analysis

## Crawling Websites and Document Conversion

Johannes Knopp, Cäcilia Zirn

Knowledge Representation and Knowledge Management Research Group
School of Business Informatics and Mathematics

13.11.2011

# Overview

1. Web Crawlers & Web Fundamentals
2. A Python Web Crawler
3. Document Conversion

# Overview

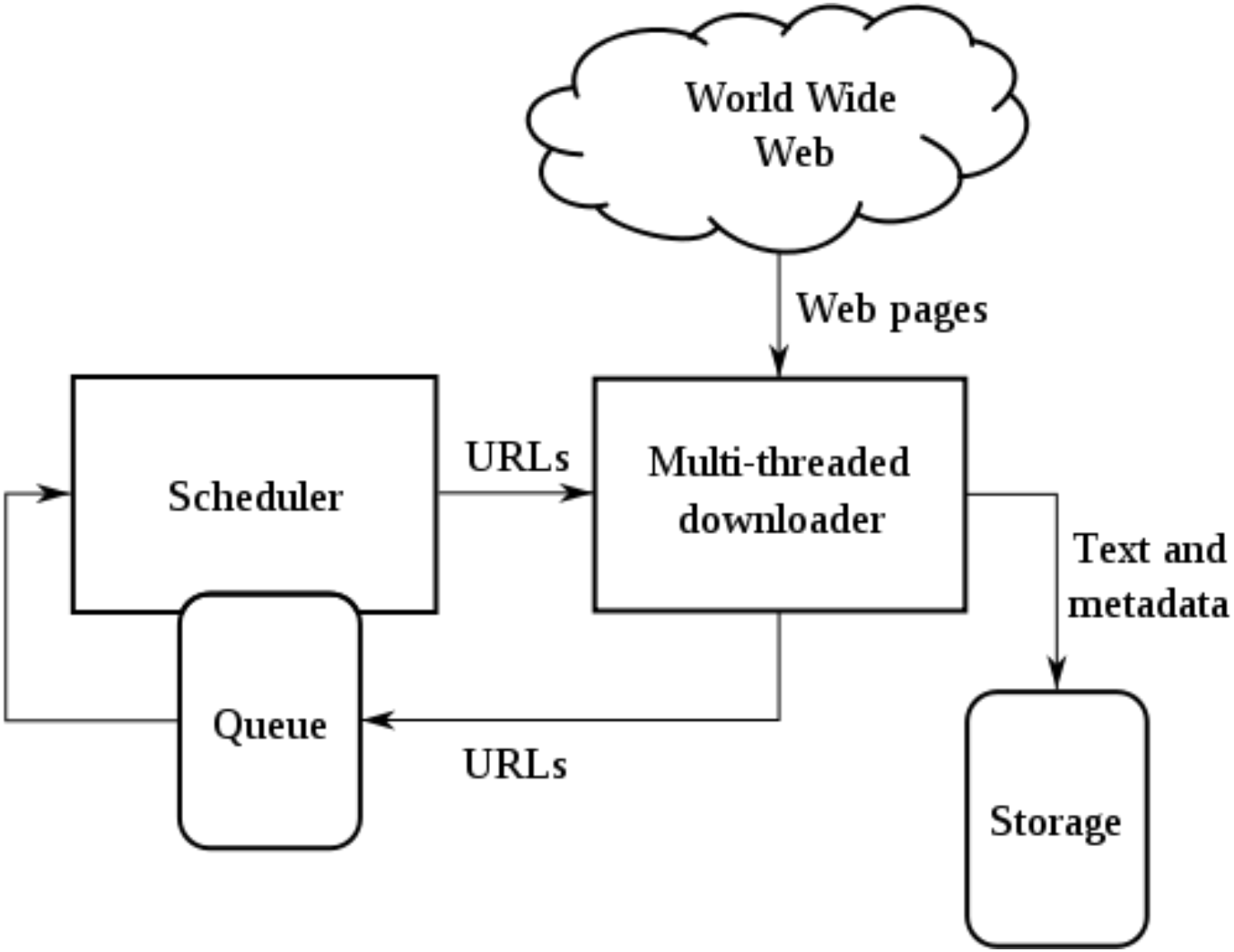
# What is a Web Crawler?

A program that automatically browses the web in an ordered way to store contents locally.

Crawlers can be used to...

- Monitor websites
- Automatically download interesting data
- Find related information by following links
- Flood your hard disk with data

# Web Crawler Architecture




http://de.wikipedia.org/w/index.php?title=Datei:WebCrawlerArchitecture.svg&oldtimestamp=20080112010717

# Hypertext Markup Language (HTML)

HTML describes how documents look like

```
<html>
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>
</html>
```

**My First Heading**

My first paragraph.

The thing within inequality signs <> is called a **tag**

# List of important tags

$< html >$ is an opening tag while $< /html >$ is a closing tag.

| **Start Tag** | **Function** | **End Tag** |
|---|---|---|
| <html> | defines the html document | $< /$html> |
| $< h1 >$,$< h2 >$,...,$< h6 >$ | heading (1=biggest) | $< /h1 >$,... |
| $< p >$ | paragraph | $< /p >$ |
| $< a$ href="link.html"> | link, goal is the value of *href* | $< /a >$ |
| $< br/ >$ | newline | |
| $< img >$ | image | $< /img >$ |
| $< table >$ | table | $< /table >$ |

## Attributes

Attributes provide additional information about an element. They are specified in the start tag and come in name/value pairs like:

```
<a href="http://www.w3schools.com">This is a link</a>
<img src="picture.jpg" width="104" height="142" />
```

Some of the standard attributes for most HTML elements:

| Attribute | Value | Description |
|---|---|---|
| class | *classname* | Specifies a classname for an element |
| id | *id* | Specifies a unique id for an element |
| style | *style_definition* | Specifies an inline style for an element |
| title | *tooltip_text* | Specifies extra information about an element (displayed as a tool tip) |

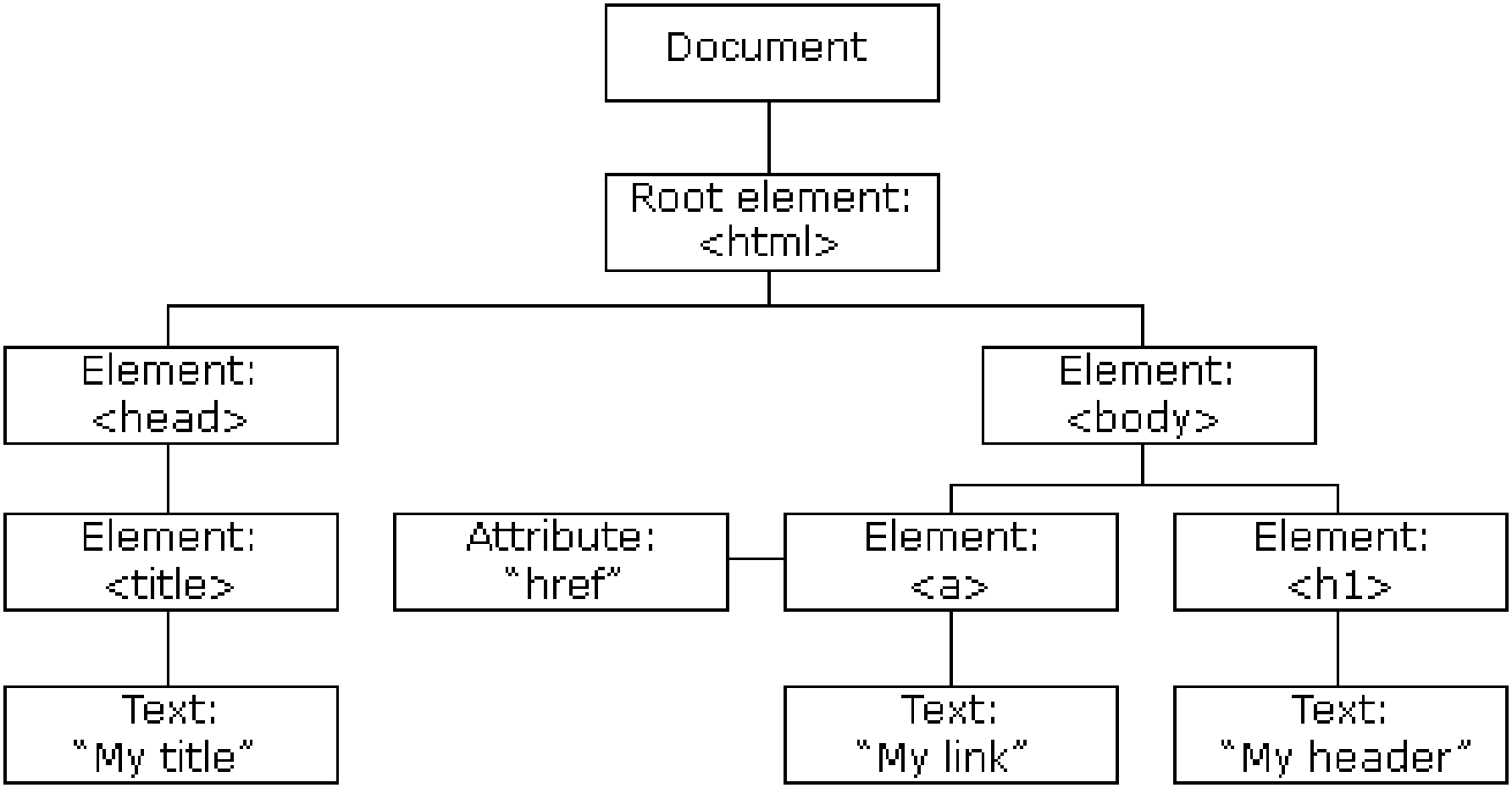
Some Elements depend on each other:

```
<html>
<body>

<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>

</body>
</html>
```

| | |
|---|---|
| row 1, cell 1 | row 1, cell 2 |
| row 2, cell 1 | row 2, cell 2 |

<tr> represents a table row; <td> the table data in the cells

# HTML forms a Tree



To learn more about HTML read this online course:

`http://www.w3schools.com/html`

# Overview

# A Python Web Crawler

Topics:

- Open websites via Python
- Access specific elements of the html (e.g. follow links)
- Download websites and documents

This part of the course is under the assumption that your goal is to download specific documents from a known website and not to crawl "the web".

# Example: 2008 USA Election Speeches

`http://2008election.procon.org/view.resource.php?resourceID=001568`

We want to download all pdfs with the candidates' speeches into folders with the respective name.
Let's develop the program together. What functions do we need?

# Open URLs

A library to work with URLs is urllib2:

```
>>> import urllib2
>>> google = urllib2.urlopen("http://www.google.com")
>>> google.read()

<!doctype html><html><head><meta
http-equiv="content-type" content="text/html;
charset=ISO-8859-1"><title>Google</title> ...
```

Save Page:

```
>>> import urllib2
>>> opened_url = urllib2.urlopen("http://www.google.com")
>>> with open("google.html","w") as output_file:
>>>     output_file.write(opened_url.read())
```

# Excursus

## **with** statement

Takes care of enter and exit routines like open and close.

```
>>> import urllib2
>>> opened_url = urllib2.urlopen("http://www.google.com")
>>> with open("google.html","w") as output_file:
>>>     output_file.write(opened_url.read())
```

roughly translates into this:

```
>>> import urllib2
>>> opened_url = urllib2.urlopen("http://www.google.com")
>>> output_file = open("google.html", "w")
>>> try:
>>>     output.file.write(opened_url.read())
>>> finally:
>>>     output_file.close()
```

# How to extract the PDFs?

Inspect the html source to find positions and regularities!

# BeautifulSoup

**BeautifulSoup** is a python library to parse html. Parsing means automatic syntax analysis and results in a parse-tree. In other words BeautifulSoup constructs the html tree we have seen earlier and allows access to any element in the tree.

`http://www.crummy.com/software/BeautifulSoup/`

```
>>> from BeautifulSoup import BeautifulSoup
>>> html = "<html><p>Para 1<p>Para 2" +
"<blockquote>Quote 1<blockquote>Quote 2"
>>> soup = BeautifulSoup(html)
>>> print soup.prettify()
<html>
 <p>
  Para 1
 </p>
 <p>
  Para 2
  <blockquote>
   Quote 1
   <blockquote>
    Quote 2
   </blockquote>
  </blockquote>
 </p>
</html>
```

```
>>> soup.findAll("p")
[<p>Para 1</p>, <p>Para 2<blockquote>Quote
1<blockquote>Quote2</blockquote></blockquote></p>]
>>> t = soup.findAll("p")[0]
>>> type(t)
<class 'BeautifulSoup.Tag'>
```

```
>>> p = '<p> <a href="foo/bar.html">link text</a></p>'
>>> p_soup = BeautifulSoup(p)
>>> p_soup.a["href"]
u'foo/bar.html'
```

# pdfminer

Converts PDFs to html, text or xml

`http://www.unixuser.org/~euske/python/pdfminer/index.html`