Web Scraping

How to make a Spider Bot


PLEASE LOVE ME
I BRINGED YOU A FLY

Web Crawlers

- A web crawler (also known as a web spider or web robot) is a program or automated script which browses the World Wide Web in a methodical, automated manner. This process is called Web crawling or spidering. Many legitimate sites, in particular search engines, use spidering as a means of providing up-to-date data. - Google

- http://www.robotstxt.org/db.html

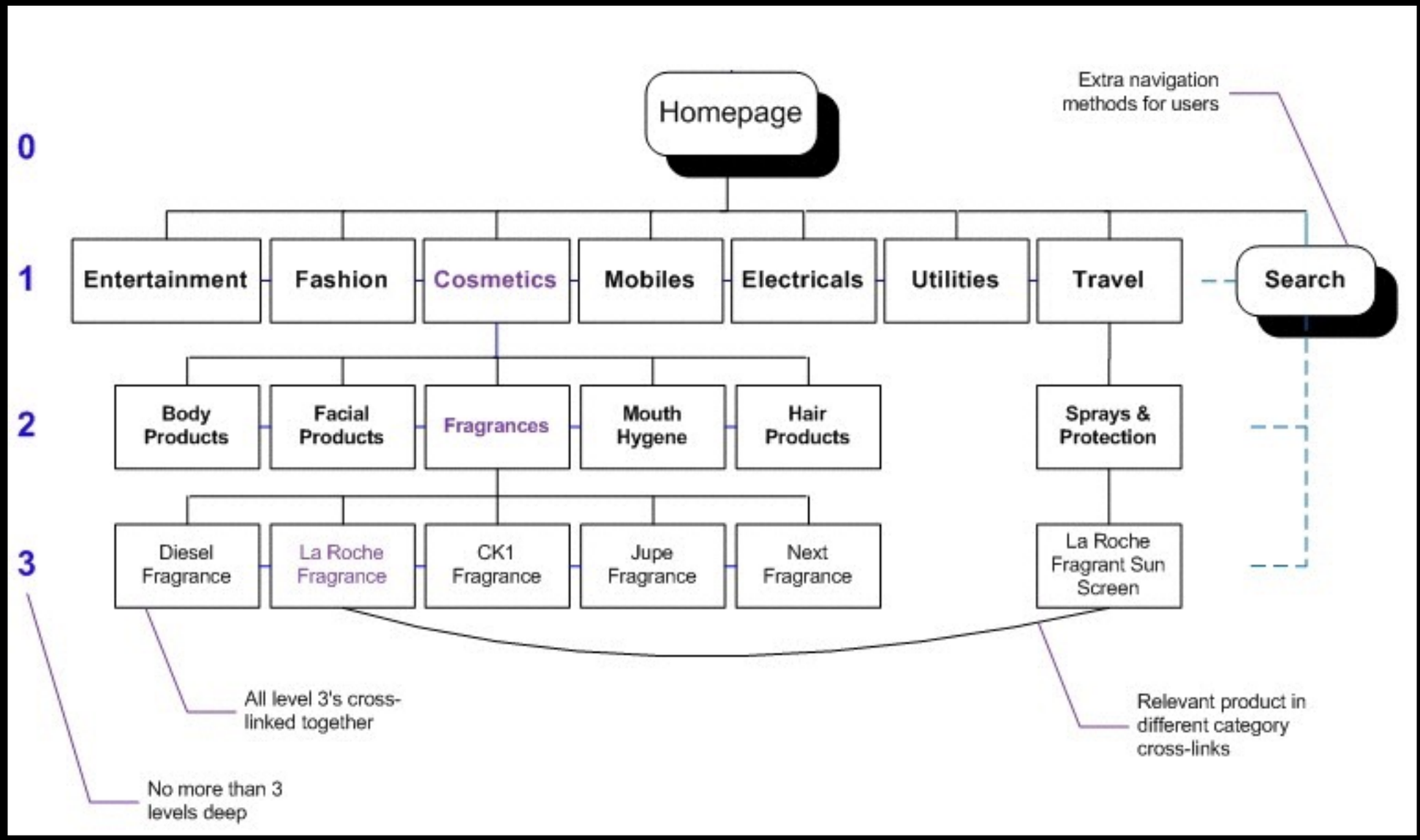
Ethics

- EULA - Respect
- Politeness Policy - don’t accidentally DDOS
- User Agent - Be honest about who you are
- robots.txt - Respect
  - http://www.robotstxt.org/

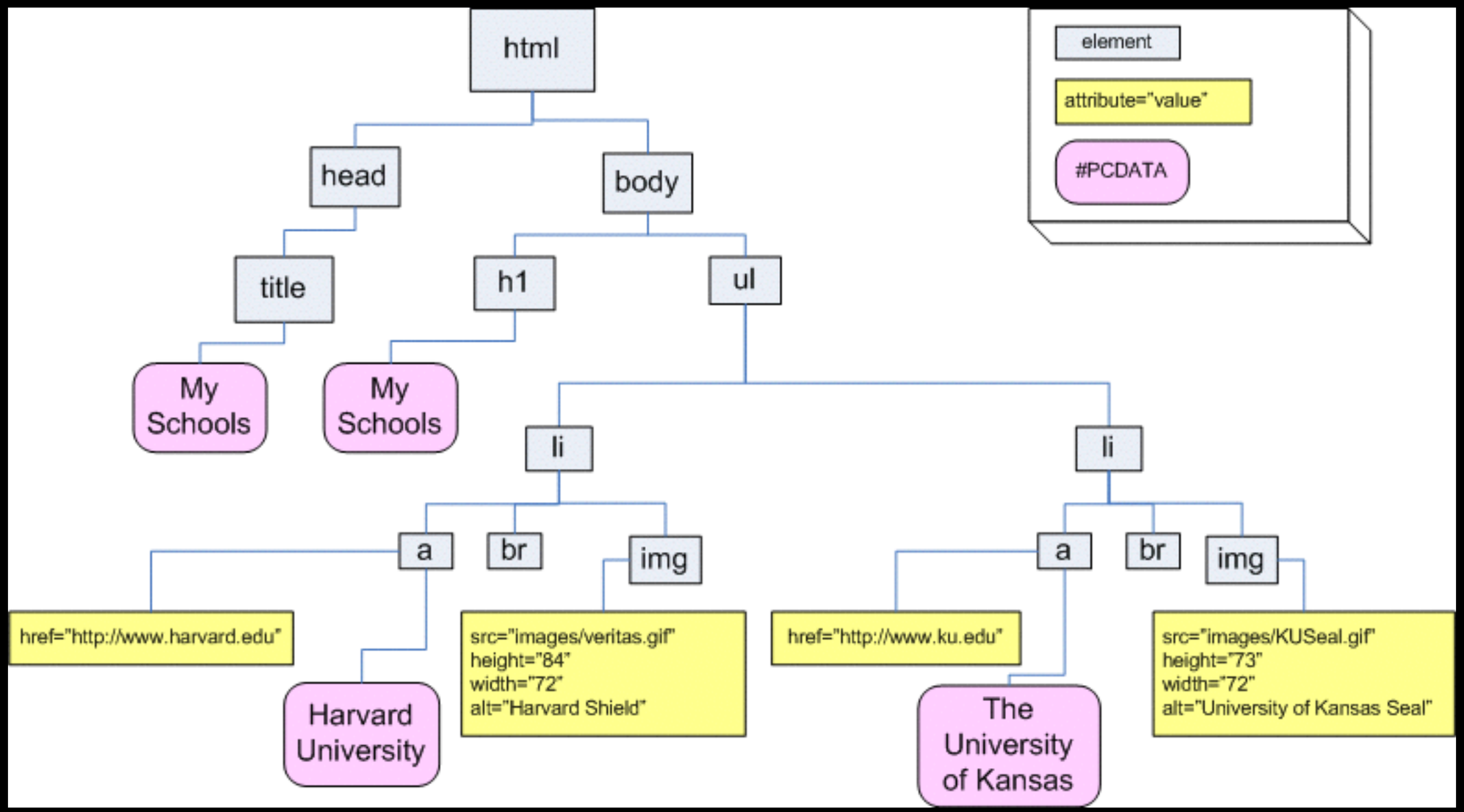
The Basics

- Get a page over HTTP
- Parse
- Process
- Follow Links
- Repeat

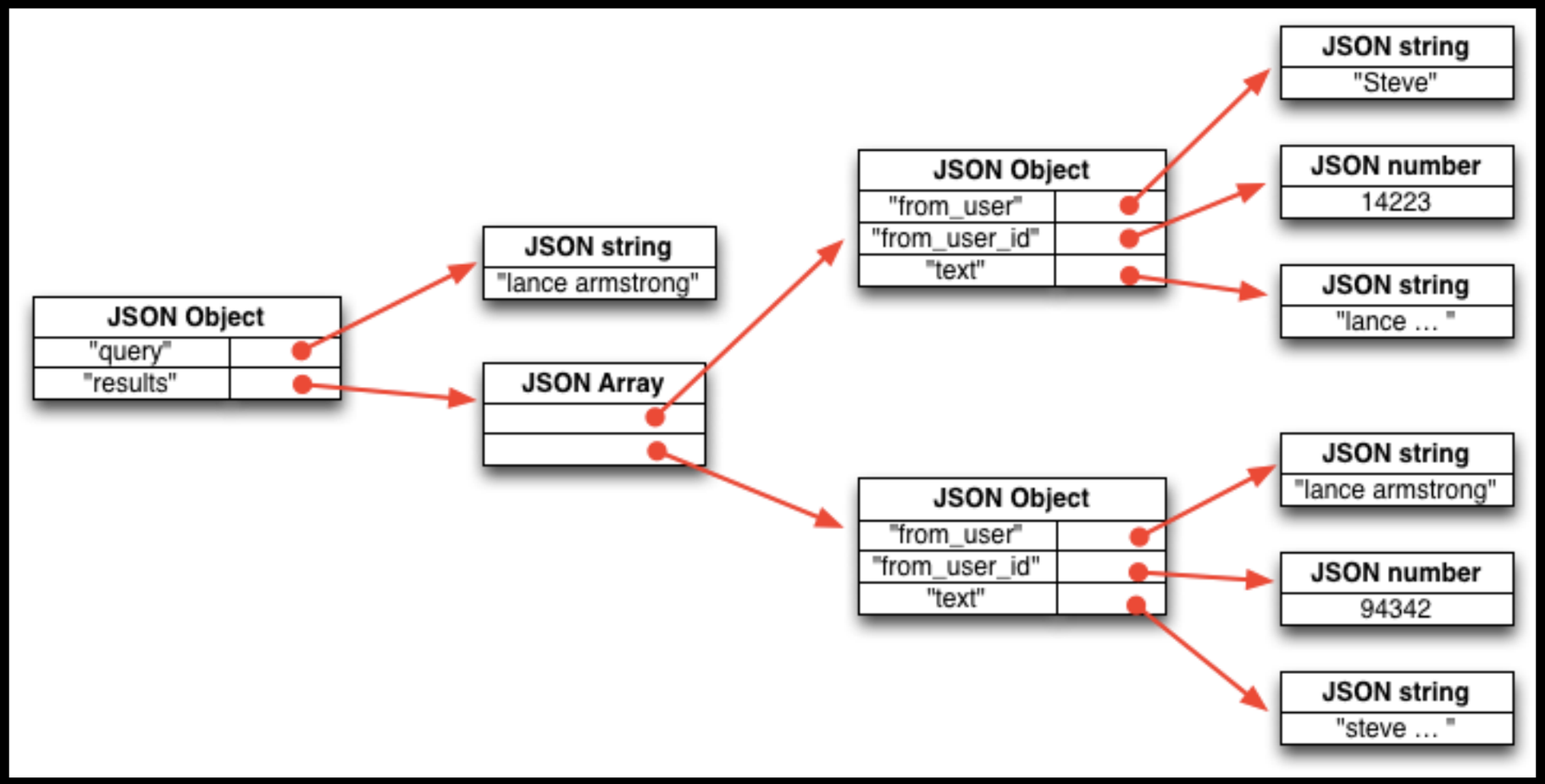
Recon:
How is your target Data Organized



Recon:
How is your target Data Organized



Recon:
How is your target Data Organized



Recon:
How is your target Data Organized

Hint: it’s probably in some kind of tree

Recon:
Lets Look At Craigslist

- I want to know the best city to buy a BMW
- Now lets design our crawler!

Step One: HTTP

- I’m using python and urllib2

Step 2: Parsing

- I am using BeautuflSoup
- `> sudo pip install beautifulsoup`

This data is for people!

and your not a people

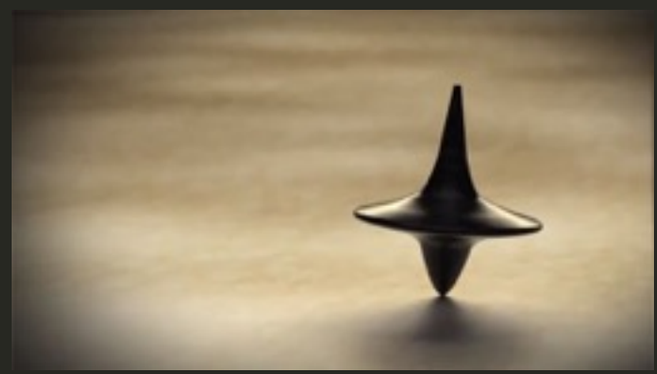
- Time for Regex Matching!

Step 3: Processing

- How do we interpret the data we found?
- “online” or “offline”?
- DIY? Or use other tools (I.E. grep?)

Recursion

```
def dfs(url, callback. depth = 1, max_depth = 10):
    """
    recursive depth-first search for links
    """

    #base case
    if depth > max_depth:
        return
    else:

    print "scraping " + url

    #HTTP get request
    root_html = urllib2.urlopen(url).read()

    callback(root_html)

    #parse and itterate over 'a' tags
    for link in BeautifulSoup(root_html, parseOnlyThese=SoupStrainer('a')):

        try:
            #find the absolute path
            next = urljoin(url, link['href'])
            dfs(next, depth + 1, max_depth)

        except Exception as e:
            print e
```



URL Normalization
and
Re-visit policy

- Avoid infinite loops

```
#ignore the inner links
if not (link['href'].startswith('#') or link['href'].startswith('?')):

    #find the absolute path
    next = urljoin(url, link['href'])

    #have we been here before?
    if not next in urls:
        #this is a new URL
```

Challenge!

- https://github.com/ufsit/webscraping-challenge12-1-2015/