

Building quick & dirty SEO Tools

A Cheat Sheet & Inspiration

Sources

APIs (more on [programmable web](#))

[AdWords](#) – Keywords

[Alchemy](#) – Structured data & text

[Bing](#) – Search, news, spelling

[Evri](#) – Sentiment and popularity

[Face.com](#) – Face detection

[Facebook](#) – Social graph

[Google Analytics](#) – Visitor data

[Hostip](#) – Geo data

[LinkedIn](#) – Professional data

[Pingdom](#) – Website uptime

Postrank ([1](#), [2](#), [3](#)) – real-time & influence

[Rapleaf](#) – Social media profiles

[Twitter](#) – Real time and social

... And of course:

[Linkscape](#) – Links

Data (more on [infochimps](#))

[Data.gov](#) – US government data

[Data.gov.uk](#) – UK government data

[Delicious list](#) – from [Peter Skomoroch](#)

[Google Public Data](#) – Directory

[Guardian](#) – content and data

[World Bank](#) – finance, health, etc.

[80legs](#) – prepackaged crawl data

Magic

YQL – [Yahoo! Query Language](#)

```
select * from html where
url="<url>" and
xpath="<xpath>"
```

```
select * from html where
url="<url>"
```

```
select * from feed where
url="<url>"
```

```
select * from search.web where
query = "<query>"
```

xpath ([more examples](#))

/foo – the element 'foo'

//bar – all elements 'bar'

foo/bar – all bar elements children of foo

foo//bar – bar arbitrary levels below foo

foo/*/bar – bar grandchildren of foo

foo/* - all children elements of foo

foo/@bar – bar attribute on foo

foo/[@bar] – foo with bar attributes

foo/[@bar=baz] – where attribute=baz

Horsepower

Crawlers / Scrapers

[Mozena](#)

[80legs](#)

[Google App Engine](#)

[Amazon Web Services](#)

Human Touch

[Amazon Mechanical Turk](#)

[Smartsheet](#) (interface to Mechanical Turk)

[oDesk](#)

Python

Since Python is the language of Google App Engine, here is how you can use YQL easily within Python:

Download [source](#) – extract to yql folder within your application

```
import yql
y = yql.Public()
result = y.execute("<yql
query>")
```

