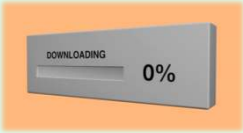
Website Scraping

In every project you need reference data – it may be a list of country names, of international phone numbers, of provinces ... Some of this data is fairly static, other data changes all the time (for instance currency exchange rates)

One way to enter data in your project is to type it in, which isn't very cool. Copying and pasting from webpages often requires a lot of typing as well (to reformat)

Sometimes, you find freely avalaible files on the web that contains reference data; for instance, files matching IP addresses to locations when you want to know where the visitors of a website are coming from (free data isn't very precise – it will tell you the country, perhaps the province. Unfortuantely, you have to pay for more precise data). This is good for data you don't update often. It's not a good solution otherwise. Some websites implement some APIs (application program interfaces), which are protocols suitable for downloading data in a format easy to understand in a program. Once again, this is often a service that you have to pay for.



The last technique, and the one that we are going to present today, consists in extracting data from web pages (which are kind of public information if not in a private member area). It doesn't always work (some web pages are populated in different stages, and your program may not always get everything), but it works often, and it's called web scraping. A scraper is this tool, and is usually associated with some long and patient work, which web scraping is in a way; but it's far better than doing it by hand. by copying and pasting from web pages.. Web scraping can be practiced in several languages (most notably python), we are going to see how to do it in Java.

Or get it from web pages...

https://jsoup.org/

The tool to use is called Jsoup, and was inspired by BeautifulSoup, a module available for Python. It's a single .jar to download. Beware that its usage is a bit different from the other .jar files we have already seen, JDBC drivers. JDBC drivers are loaded by reflection at load time, not jsoup. In practice, it means that you must know where to find it (CLASSPATH) not only when you run the program, but also when you compile it.

```
<html>
  <head>
  </head>
  <body>

  </body>
</html>
```

How does Jsoup work? First of all, we need to understand how a HTML file is built. Notice that indentation and carriage returns are here for clarity, but are irrelevant in a HTML page. There is a head section that doesn't appear on the page and is irrelevant for scraping. What we'll take a look at is the body, which is what the browser displays. In this body, many tags. The most relevant ones when scraping are probably div, span and table (plus tr, th, td and so forth), but it's not exclusive. One important point is that tags are often nested.

```
<div></div>
<span></span>
<table></table>
```

```
<html>
  <head>
  </head>
  <body>

  </body>
</html>
```

Tags often have attributes, associated with the opening tag. A commonly used tag is "class", which defines a category of tags that have a common appearance or way of reacting to interaction.

```
<div class="name"></div>

<span></span>

<table></table>
```

```
<html>
  <head>
  </head>
  <body>

  </body>
</html>
```

id indicates an identifier, which should normally be unique inside the page and can help locate a particular section.

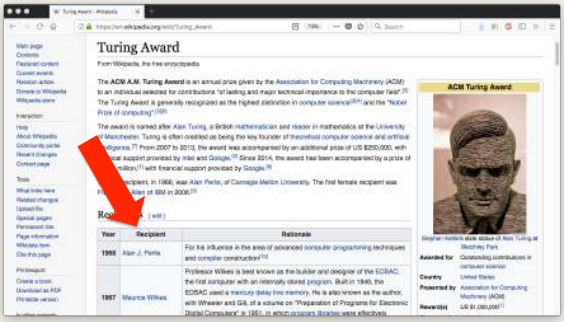
```
<div class="name"></div>

<span id="name"></span>

<table></table>
```

Using Jsoup

Jsoup class is for Connection, Document is the whole page, Element is an element but we can get several (list) so we need Elements (a collection)

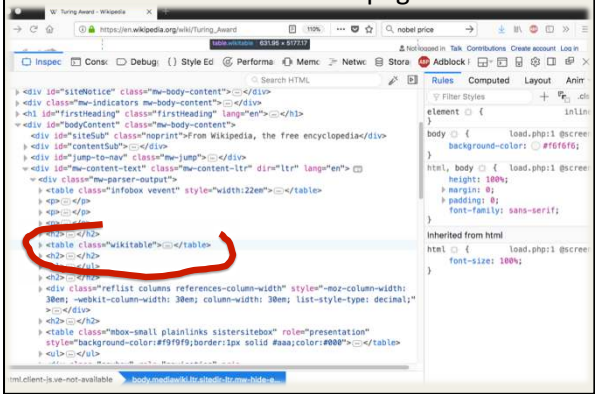
```
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;
import org.jsoup.helper.Validate;

import java.io.IOException;


Document doc = Jsoup
      .connect("https://en.wikipedia.org/wiki/Turing_Award")
      .get();
```

But what do we want in the page?



But what do we want in the page?

But what do we want in the page?



Using Jsoup

```
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;
import org.jsoup.helper.Validate;

import java.io.IOException;


Document doc = Jsoup
      .connect("https://en.wikipedia.org/wiki/Turing_Award")
      .get();


Elements tables = doc.select("table.wikitable");
```

```
for (Element t: tables) {
   Elements rows = t.select("tbody tr");
   for (Element r: rows) {
      Elements cells = r.select("td");
      if (cells.size() > 0) {
         Elements year = r.select("th");
         System.out.print(year.get(0).text() + ",");
         System.out.println("\""+cells.get(0).text()+"\"");
      }
    }
  }
```

```
<tr>…</tr>
<tr>
  <th>1976</th>
  <td>
    <a href="/wiki/Michael_O._Rabin" title="Michael O. Rabin">Michael O. Rabin</a>
    and
    <br>
    <a href="/wiki/Dana_Scott" title="Dana Scott">Dana S. Scott</a>
  </td>
  <td>…</td>
</tr>
<tr>…</tr>
<tr>…</tr>
<tr>…</tr>
```

```
Elements winners = cells.get(0).select("a");
for (Element w: winners) {
   System.out.print(year.get(0).text() + ",");
   System.out.println("\""+w.text()+"\"");
}
```

Last Stage in the Project

* Updated database posted
* On startup, check database status

```
select max(id),max(UTC_date)
from quakes;
```

* Collect the new quakes from

https://www.emsc-csem.org/Earthquake/?view=1

Screen number

Inserting a new quake

Must extract N/S from latitude, E/W from longitude

SQL:

```
insert or ignore into quakes(id,UTC_date,
             latitude,longitude,
             depth,magnitude,region)
values(?,?,
       case ? when 'N' then 1 else -1 end * ?,
       case ? when 'W' then -1 else 1 end * ?,
       ?,?,?)
```

Every parameter can be passed with **`setString()`**
(converted automatically)

`area_id` is automatically set by a mechanism called *trigger*