Taller #8: Web Testing

Generación Automática de Tests

CrawlJax

- Explorar aplicaciones AJAX
- Opensource:http://crawljax.com
- Estado: DOM Tree
- Eventos: click, mouseover, dbclick… sobre HTML elements
- Compara el DOM Tree **antes** y **después** de cada evento (Levenshtein edit distance)

Invariantes

- Sobre un único estado (DOM tree)
  - Validación del DOM, Mensajes de error, accesibilidad,
- Entre estados
  - Back-Button consistente, No clicks “muertos”
- User-defined:
  - Escribir un predicado que se evalúa en cada nuevo estado visitado

BabyCrawler

- Implementar un “baby" crawler de páginas web para un cierto dominio (ej: www.dc.uba.ar/*)

- Una página que ya haya sido visitada no debe ser visitada dos veces.

- Sólo si la página pertenece al dominio, hay que explorar todos sus links.

https://jsoup.org

jsoup | News | Bugs | Discussion | Download | API Reference | Cookbook | Try jsoup

jsoup » jsoup: Java HTML Parser

jsoup: Java HTML Parser

`jsoup` is a Java library for working with real-world HTML. It provides a very convenient API for extracting and manipulating data, using the best of DOM, CSS, and jquery-like methods.

`jsoup` implements the WHATWG HTML5 specification, and parses HTML to the same DOM as modern browsers do.

- scrape and parse HTML from a URL, file, or string
- find and extract data, using DOM traversal or CSS selectors
- manipulate the HTML elements, attributes, and text
- clean user-submitted content against a safe white-list, to prevent XSS attacks
- output tidy HTML

jsoup is designed to deal with all varieties of HTML found in the wild; from pristine and validating, to invalid tag-soup; jsoup will create a sensible parse tree

Cookbook contents

```
@Test
public void testGetPage() throws IOException {
    String USER_AGENT = "Mozilla/5.0 (Windows NT 6.1; WOW64)";
    USER_AGENT += " AppleWebKit/535.1 (KHTML, like Gecko)";
    USER_AGENT += " Chrome/13.0.782.112 Safari/535.1";
    String url = "https://www.dc.uba.ar/";
    Connection connection = Jsoup.connect(url).userAgent(USER_AGENT);

    Document htmlDocument = connection.get();
    int statusCode =  connection.response().statusCode();
    String mimeType = connection.response().contentType();
    String bodyText = htmlDocument.body().text();
    Elements linksOnPage = htmlDocument.select("a[href]");
    Element aLink = linksOnPage.iterator().next();
    String aUrl = aLink.absUrl("href");
}
```

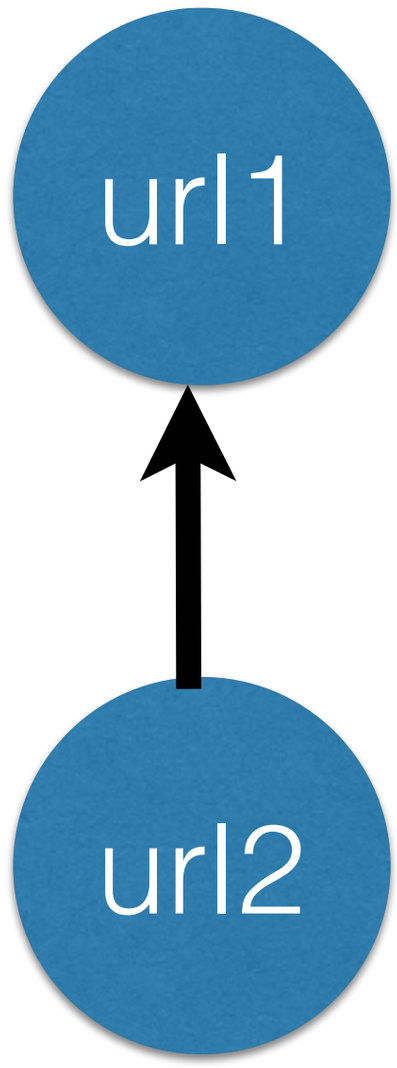
Connection.get()

- **UnsupportedMimeTypeException**: El Documento no posee texto html (ej. pdf, jpg, png, etc.)
- **MalformedURLException**: el URL no es un URL válido
- **HttpStatusException**: 4xx client errors, 5xx server errors
- **SocketTimeoutException, UnknownHostException, SSLHandshakeException, etc.**

Grafo de navegación del sitio web

- Un grafo es una estructura de datos que tiene nodos y arcos.

- En este caso, casa nodo es un URL y un arco es un link que está en una página para acceder a otra



BabyCrawler

- Completar el código para que **BabyCrawler** recorra todos los links **de un dominio url** (sin repetir la visita a una página)

- El **BabyCrawler** debe completar toda la información del modelo del sitio web (clase **WebSite**)

- Usar la clase **Logger** para mostrar por consola el avance del crawler.

Ejercicio #1

- Crawler el sitio https://sbst2017.lafhis.dc.uba.ar/
- ¿Cuántos links internos hay?
- ¿Hay algún link roto?

Ejercicio #2

- Crawler el sitio https://icc.fcen.uba.ar/
- ¿Cuántos links internos hay?
- ¿Hay algún link roto? ¿Cuáles? ¿En cuántas páginas son usados?
- ¿Qué otras IO Exceptions encontró?

Ejercicio #3

- Crawler el sitio https://www.dc.uba.ar/
- ¿Cuántos links internos hay?
- ¿Hay algún link roto? ¿Cuáles? ¿En cuántas páginas son usados?
- ¿Qué otras IO Exceptions encontró?