OXPath: Everyone can Automate the Web![*]

Tim Furche, Georg Gottlob, Giovanni Grasso, Christian Schallhart
Department of Computer Science, Oxford University, Wolfson Building, Parks Road, Oxford OX1 3QD
firstname.lastname@cs.ox.ac.uk

1. WHY: AUTOMATING WEB ACTIONS

Web data easily accessible to everyone is the Holy Grail 2.0. Scientists need data to study, e.g, how people interact on web social networks, whereas web companies use profiling data to target online ads or improve search results, and quantitative analyst examine streams of events to predict market variations. We all face daily tasks (e.g., planning holidays or searching for a new camera), for which web data (e.g, reviews) plays an important role. In principle, all the necessary information is readily available on some web page, yet manually accessing, extracting, and aggregating that information is often infeasible due to the number of different sites and the size of the involved data. This creates a new divide in data-driven research and analysis between governments or large, web-savvy companies that can exploit web data at scale and most other entities or persons that do not have that ability.

Web data extraction addresses the problem of turning data accessible through existing, human-oriented interfaces, into structured data. For instance, each gray `span` HTML element with CSS class `source` on Google News should be recognized as news source. However existing tools for web data extraction are either research prototypes not fit for everyday users or very expensive, commercial applications that require significant resources for large scale data extraction. Furthermore, they are usually not designed with end users in mind, as commercial data extraction is primarily offered as a service these days. Data extraction tools are also quickly outpaced by the growth and change in web technologies.

2. WHAT: OXPATH

Therefore, we have introduced OXPath as a new generation tool for scalable data extraction and automation. It builds on XPath, an established, standard technology in the web, and is developed as an open source tool by an international community. Furthermore, we are currently developing a suite of end user tools that allow OXPath to be used by everyone regardless of the technical background, including visual wrapper generator and easy cloud-based extraction. Of course, large scale data extraction will always require some technical veracity, e.g., for storage, cleaning, and analysis of the data. But with OXPath, we aim to make the use of web sites as a data source just as easy as using a local database. More specifically, OXPath extends XPath with only 4 concise extensions, yet provides all that is necessary to deal with modern web applications:

(1) OXPath allows the *simulation of user actions* (e.g., click, form filling) to interact with the scripted interfaces of web applications.

(2) In addition to the selection capabilities of XPath, OXPath allows *selection based on visual features* by exposing all *CSS properties*. It is possible, e.g., to select elements by their color.

(3) OXPath deals with navigation through page sequences (*multi-way navigation*, e.g., following multiple links from the same page, and *unbounded navigation sequences,* e.g., following next links on a result page until there is no further one.

(4) OXPath enables the identification of *data for extraction,* which can be assembled into (hierarchical) records, regardless of its original HTML structure.

All these features are achieved without sacrificing performance: OXPath scales well both in time and in memory and uses very little resources compared to other web data extraction tools. Specifically, its *memory requirements are independent* of the number of pages visited. To the best of our knowledge, OXPath is the first web extraction tool with such a guarantee [1]. Low resource usage is crucial for use as an open information access tool, as it directly translates into low cost for cloud extraction.

As a basic example, consider the following OXPath expression.

```
doc("news.google.com")//div[@class~="story"]:<story>
 [.//h2:<title=string(.)>]
 [.//span[style::color="#767676"]:<source=string(.)>]
```

It navigates on Google News, and extracts a `story` element for each current news story on the page, along with its title and sources (selected by color), producing:

```
<story><title >Tax cuts ...</title>
       <source>Washington Post</source>
       <source>Wall Street Journal</source> ... </story>
```

We provide an open source OXPath implementation available at `http://diadem-project.info/oxpath`. Further, to support users not familiar with OXPath, we have developed a visual tool to assist building extraction tasks. Using Visual OXPath [2], without any knowledge of OXPath, users can develop robust extraction expressions just with few clicks. Given only one example, (e.g., one story on Google News), the tool exploits similarity to automatically suggest the expression that selects all the stories on that page.

3. HOW: DEMONSTRATION EXAMPLE

To extract the most popular **petitions on "Government&Politics"** listed on `petitionspot.com`, a user has to perform the following sequence of actions to retrieve the page listing these petitions (see

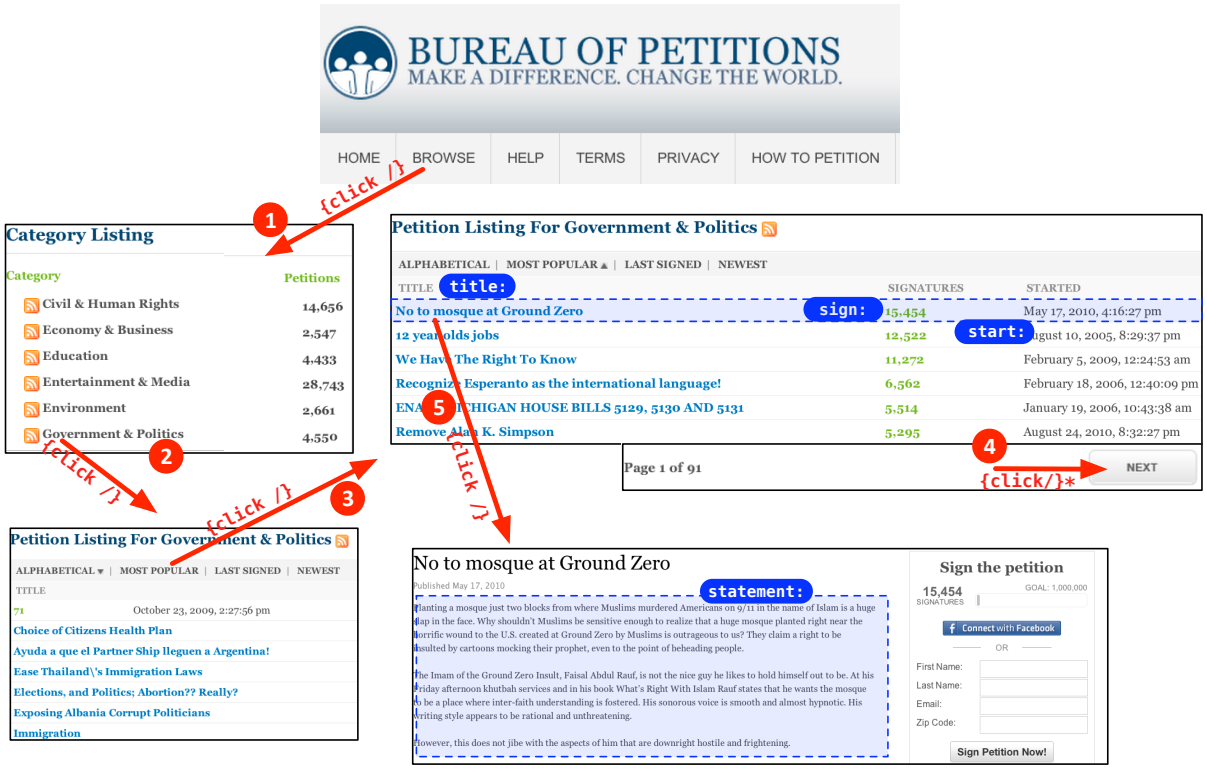

Figure 1: Finding an OXPath through Petition.com

Figure 1): **(1)** Click "Browse" on the header menu, **(2)** click on the link "Government&Politics" , and finally **(3)** click on the link "Most Popular" to order the petitions by popularity. Then, for each petition on that page, **(4)** click on its title to reach the details page for retrieving the full statement. Finally, **(5)** click on the "Next" link and repeat the same actions on the following pages.

The following is an OXPath expression that realizes this task, extracting relevant data for each petition.

```
doc("http://www.petitionspot.com/")//div#navigation//li[2]/a[1]/{click/}❶
2 /(//a.gray)[6]/{click/}❷
  //a[starts-with(.,'Most Popular')]/{click/}❸
4 (//a/img[@alt='Next']{click/})*❹
  //div.pad5/div:<petition>
6         [.//a:<title=string(.)>/{click/}❺
                  //div#p_content:<statement=string(.)>]
8         [.//div.green:<signatures=string(.)>]
          [.//div.col_started:<start=string(.)>]
```

To identify the "Browse" link (Line 2), we adopt the `#` notation from CSS for selecting elements (`div`) with an `id` attribute `navigation`. On our example, this identifies the header navigation menu. Upon that, in step **(1)** we `click` on the second link (`li[2]/a`), to reach the following page. Step **(2)** (line 3) clicks on the sixth link having a CSS class gray (`a.gray`), whereas on the returned page, we `click` on the link whose text starts with "Most Popular" **(3)**.

Before extracting the relevant data for each petition on the current page, we instruct OXPath to iterate on all following pages, to perform an exhaustive extraction. To this end, OXPath introduces the Kleene star `(path)*` operator, to "repeat path until it matches". In our example, we continue clicking on the the "Next" link (step **(4)**, line 5), until any further is found.

On each page, we can extract the relevant data as follows. OXPath allows labelling data for extraction through *extraction markers*. We first identify petition boundaries as the `div` element with class `pad5`. We label these records as `petition` using the record extraction marker `:<petition>`. From there, we navigate to the contained title links and extract their value as a `title` (`:<title>`) attribute, and `click` (step **(5)** on the link to obtain the page for the individual petition, where we find and extract its full `statement`. Finally, we extract the number of signatures and start date from the previous page – without caring for the order in which the pages are visited during extraction. OXPath buffers pages when necessary, yet guarantees that the number of buffered pages is independent of the number of visited pages.

It is worth emphasizing, that this example expression can be generated in visual OXPath by performing the form filling once, selecting one example for each attribute, and identifying the next link. For all that, no knowledge of XPath or OXPath is required.

Acknowledgements

The research leading to these results has received funding from the European Research Council under the European Community's Seventh Framework Programme (FP7/2007–2013) / ERC grant agreement DIADEM, no. 246858.

4. REFERENCES

[1] T. Furche, G. Gottlob, G. Grasso, C. Schallhart, and A. Sellers. Oxpath: A language for scalable, memory-efficient data extraction from web applications. In *Proc.Int'l. Conf. on Very Large Data Bases (VLDB)*, 2011.

[2] J. Kranzdorf, A. Sellers, G. Grasso, C. Schallhart, and T. Furche. Spotting the tracks on the oxpath. In *International Conference on Word Wide Web (WWW 2012), (Companion Volume)*, 2012.