WAIL: Collection-Based Personal Web Archiving

John A. Berlin, Mat Kelly, Michael L. Nelson, Michele C. Weigle
Old Dominion University, Department of Computer Science, Norfolk VA, 23529, USA
{jberlin,mkelly,mln,mweigle}@cs.odu.edu

ABSTRACT

Web Archiving Integration Layer (WAIL) is a desktop application written in Python that integrates Heritrix and OpenWayback. In this work we recreate and extend WAIL from the ground up to facilitate collection-based personal Web archiving. Our new iteration of the software, WAIL-Electron, leverages native Web technologies (e.g., JavaScript, Chromium) using Electron to open new potential for Web archiving by individuals in a stand-alone cross-platform native application. By replacing OpenWayback with PyWb, we provide a novel means for personal Web archivists to curate collections of their captures from their own personal computer rather than relying on an external archival Web service. As extended features we also provide the ability for a user to monitor and automatically archive Twitter users' feeds, even those requiring authentication, as well as provide a reference implementation for integrating a browser-based preservation tool into an OS native application.

KEYWORDS

Personal Web Archiving

ACM Reference format:
John A. Berlin, Mat Kelly, Michael L. Nelson, Michele C. Weigle. 2017. WAIL: Collection-Based Personal Web Archiving. In *Proceedings of Joint Conference on Digital Libraries, Toronto, Ontario, Canada, June 2017 (JCDL'17),* 2 pages.
DOI: 10.XXX/XXXX

1 INTRODUCTION

Subscription-based Web archiving services like Archive-it allow users with limited technical knowledge to create and replay personalized collections of Web archives. Archive-It provides its users with a simple interface to create collections and to launch complex archival crawl. Similar to Archive-It is Webrecorder[1], which allows any user to register for the service and provides them with the ability to create and manage personalized collections of Web archives. But unlike Archive-It, Webrecorder requires its user to manually drive the preservation process or upload content for replay while only providing its users up to five gigabytes of storage. Individuals that wish to freely (*gratis* and *libre*) archive Web pages without arbitrary restrictions beyond the limitations of their personal computers using institutional grade tools must setup an archival Web crawler (e.g., Heritrix) and replay system (e.g., Wayback), time consuming and technical tasks potentially beyond the individual's skill level. Even if a user is able to successfully set up these tools, they must also configure the crawls via Heritrix and come up with their own means of associating the Web archives to each other for collection-based replay via Wayback.

We have developed a tool that provides users with access to both Heritrix and Wayback while providing an interoperable mechanism for personal collection-based Web archiving from their personal computers. Users can create and add to these collections through WAIL-Election with the software taking care of the details in managing the collections, crawls, and replay. We have integrated a native Chromium[2] browser (the core of Google's Chrome Web browser) into the archival process in order to surface content specific to sites like Twitter for more accurate and comprehensive preservation.

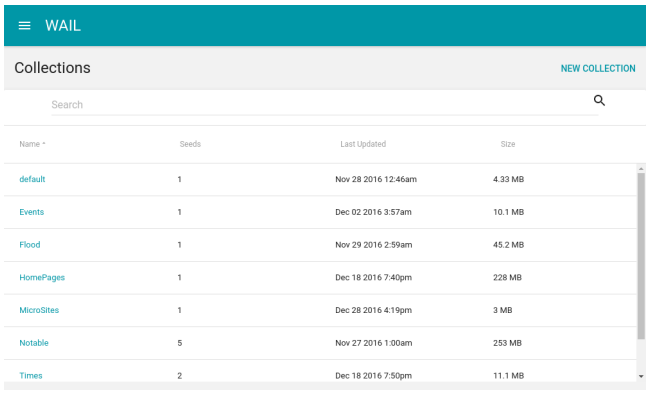

Figure 1: Collections screen

2 WAIL

WAIL-Electron is an Electron[3]-based reimplementation of the original WAIL application [5]. WAIL-Electron (from here on referred to as "WAIL") allows users to create and manage personalized collections of Web archives from their personal computers. When a user first starts the application, WAIL provides them with a default collection and the means to create additional collections straight away from the collection screen (Figure 1). The collection view displays an overview of the collections WAIL is currently managing and information about them. This information includes the number of seeds contained in the collection along with the collection's size and the last time it was updated. A user may easily create a new collection by clicking the "New Collection" button.

Doing so displays a dialog (Figure 2), prompting the user for a collection name, title, and description. These values are propagated to the WAIL interface and are viewable when replaying the collection through Wayback. When viewing a collection, WAIL displays

[1]https://webrecorder.io/

[2]https://www.chromium.org/
[3]http://electron.atom.io/

JCDL'17, Toronto, Ontario, Canada

DOI: 10.XXX/XXXX

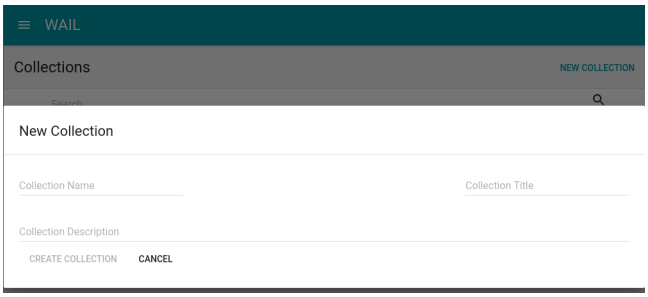

Figure 2: Create Collection

metadata about each seed in the collection like date added, the last time archived, the number of captures and direct means to replay the seed using the Wayback instance provided. Users may also add seeds to be crawled through the WAIL interface or may import any previously existing WARC [3] files that were generated from any source. WAIL also provides a mechanism to check on the state of the resource on the live Web from within the same interface prior to the user initiating a crawl.

After entering the URI for a new seed, WAIL automatically configures and launches the crawl. While a crawl or a set of crawls is underway, WAIL provides real-time progress monitoring for the crawls within the interface. Upon completion of a crawl, WAIL will automatically associate the generated WARC file to the collection and ensure its ingestion by Wayback. As an additional feature, WAIL provides an interface for a user to monitor and archive Twitter content automatically. Users may specify criteria by which to identify tweets to be archived and to which collection the preserved tweet is added. Once the monitoring has started and a tweet has been identified for archiving, the tweet will be archived and automatically added to the specified collection.

3 IMPLEMENTATION

We implemented the new version of WAIL in Electron, as it allows the application to be more consistent across the Mac, Windows, and Linux platforms through its use of an embedded Node.js[4] and a native Chromium (Chrome) browser. Since Electron applications are written entirely using JavaScript, HTML, and CSS; special consideration had to be taken as to how WAIL would handle the scale required for such an application. Instead of using threads, as would be done in a Python or Java application, WAIL leverages browser windows with Electron's Node.js integration to achieve the same effect. Each window is responsible for the file system and OS native process-intensive tasks required for the collection and crawl management functions of WAIL. Unlike our previous revision of WAIL, where we used OpenWayback[5] as the replay engine, we now use PyWb[6] for replay and to provide the base collection functionality.

As an additional feature, we leveraged Electron's Chromium foundation by integrating the functionality of our previously created browser extension, WARCreate [4], into the native application archival process. We begin the preservation by loading a URL into a `webview`[7], which allows live Web content to be embedded into the application. Unlike an `iframe`, the webview tag runs the guest page in a separate process without the same permissions as the application and exposes the Web contents of the guest page. Utilizing the Web contents, we monitor and capture the HTTP requests and responses made while rendering the page. When the page has been determined to be fully rendered (loaded), we extract the HTML of the page and generate a WARC file. This file is then added to the collection specified by the user.

4 EVALUATION

We tested our browser archival implementation by comparing it with a Heritrix crawl of the WS-DL Twitter (@WebSciDL) feed. The Heritrix crawl was configured to only crawl the page and all of its embedded resources to ensure as close of a comparison as possible. The Heritrix crawl took 9 minutes 45 seconds and made 208 requests, inclusive of the mobile representation of the page. Using our in-browser implementation, the page was archived in 1 minute 30 seconds with 53 requests. These requests represent the actual requests made by the browser when parsing the representation as well as all requests made from JavaScript within the target page. The additional requests made by Heritrix can be attributed to how Heritrix contructs its crawl frontier [2]. Heritrix extracts any URL it can identify from the embedded resources [6] of the representation adding them to the crawl frontier.

5 FUTURE WORK

Unlike headless browsers like PhantomJS, we have built the preservation mechanism directly into the browser. We would like to use this to begin preserving streaming media and advertisements that are often problematic to preserve using conventional means. Brunelle *et al.* [1] showed the impact of JavaScript on archivability of Web pages and we believe the JavaScript execution gained through using a full browser would allow for this content to be better preserved.

6 ACKNOWLEDGEMENTS

This work is supported by the National Endowment for the Humanities (NEH), through Digital Humanities grants HD-51670-13 and HK-50181-14.

REFERENCES

[1] Justin F Brunelle, Mat Kelly, Michele C Weigle, and Michael L Nelson. 2016. The Impact of JavaScript on Archivability. *International Journal on Digital Libraries* 17, 2 (2016), 95–117.
[2] Justin F. Brunelle, Michele C. Weigle, and Michael L. Nelson. 2016. *Adapting the Hypercube Model to Archive Deferred Representations and Their Descendants.* Technical Report arxiv:1601.05142.
[3] ISO 28500. 2009. WARC (Web ARChive) file format. http://www.digitalpreservation.gov/formats/fdd/fdd000236.shtml. (August 2009).
[4] Mat Kelly and Michele C. Weigle. 2012. WARCreate - Create Wayback-Consumable WARC Files from Any Webpage. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries (JCDL).* 437–438.
[5] Mat Kelly, Michele C. Weigle, and Michael L. Nelson. 2013. Making Enterprise-Level Archive Tools Accessible for Personal Web Archiving. Personal Digital Archiving, poster. (February 2013).
[6] Hunter Stern. 2011. Fetch Chain Processors. https://webarchive.jira.com/wiki/display/Heritrix/Fetch+Chain+Processors. (2011).

[4] https://nodejs.org/en/
[5] http://www.netpreserve.org/openwayback
[6] https://github.com/ikreymer/pywb
[7] http://electron.atom.io/docs/api/webview-tag/