WebMacros - a Proxy-based System for Automating User Interactions with the Web

Alex Safonov
University of Minnesota
4-192 200 Union St SE
Minneapolis, MN 55455
1-612-626-8396
safonov@cs.umn.edu

Joseph A. Konstan
University of Minnesota
4-192 200 Union St SE
Minneapolis, MN 55455
1-612-625-4002
konstan@cs.umn.edu

John V. Carlis
University of Minnesota
4-192 200 Union St SE
Minneapolis, MN 55455
1-612-625-4002
carlis@cs.umn.edu

ABSTRACT

WebMacros is a proxy-based system for automating repetitive user interactions with the Web by recording and replaying user navigation. The innovations in the system include its ability to compare HTML pages based on structure, which is used to verify the correct playback, and the ability to remotely execute and share macros.

Keywords

Macro, demonstration, playback, sharing, proxy

1. INTRODUCTION

Web technologies including POST forms, dynamically generated pages, cookies, expiring session tokens etc., have made traditional bookmarks and histories insufficient for managing users' personal views of the Web. Many pages do not have a well-defined URL, but require a sequence of steps to be retrieved; these were termed hard-to-reached pages [1]. Examples of hard-to-reach pages include citation engine searches and airfare availability queries. The need to automate repetitive user interactions with the Web motivated us to develop WebMacros [4], a system for demonstrating and playing user actions on the Web.

2. WEBMACROS FEATURES

2.1 Creating macros by recording user actions

A user initiates recording of a macro explicitly, by navigating to the WebMacros controls page (its URL can be bookmarked or placed in a browser toolbar), and pressing the "Start Recording" button. She then demonstrates the macro by the normal process of Web navigation and form filling. Page retrieval speeds are comparable to those when no recording occurs. When all the steps are demonstrated, the user presses the "Stop Recording" button and is prompted to enter macro name and description.

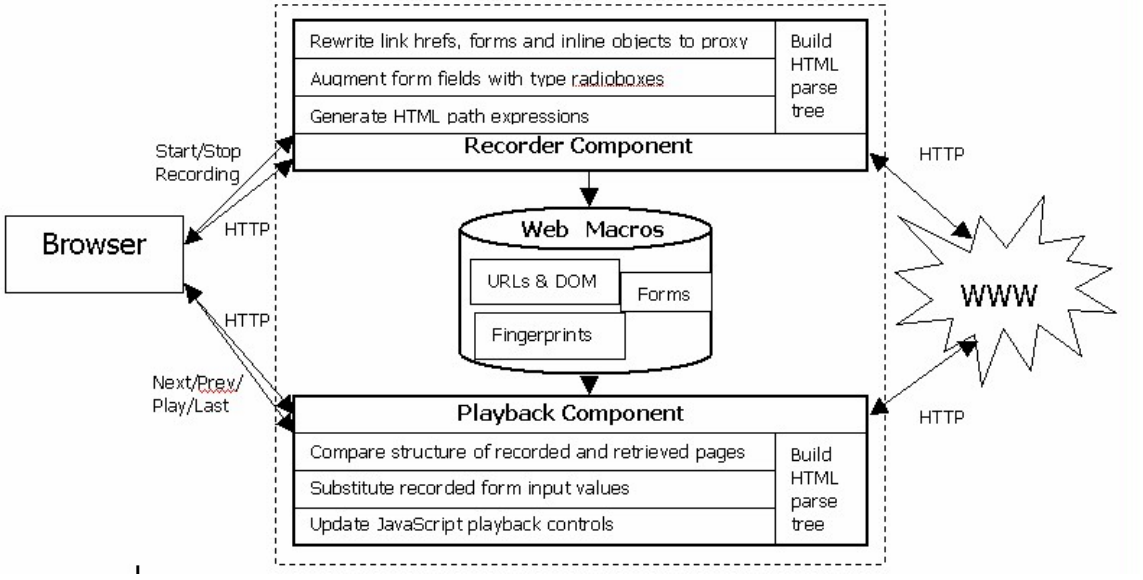
2.2 Support for parametric macros

As a user demonstrates a macro, she can specify the type of INPUT and SELECT form elements by clicking on radioboxes added by the recording system. The following parameter types are supported: private (default for PASSWORD and inferred username fields), constant (default for all other fields), and variable. During playback, recorded values of constant and variable fields are substituted into each retrieved page; constant fields cannot be overridden by the user.

2.3 Verifying pages retrieved at playback

Since each macro step can at playback retrieve an unexpected page (such as a login failure, or no results matching a query), the system needs to compare the retrieved pages against the recorded ones. However, verbatim comparison of HTML content is insufficient for many dynamically generated pages, since a server can populate the same HTML template with different results based on current data and playback parameters.

To compare pages based on structure rather than content, WebMacros represents them as sets of path expressions in the HTML parse trees. Our experiments on eBay! and Amazon.com data [5] indicate that a simple page structure similarity measure reliably classifies pages into types, and can be used to verify results of WebMacros playback.



2.4 Interactive and batch macro playback

A user can replay a recorded macro either in a batch mode, in which the system loads in the browser only the last retrieved page, or in an interactive mode. Interactive mode is useful when the user is interested in pages produced by intermediate macro steps. WebMacros provides a JavaScript-based playback control panel with the Prev, Next, Play, and Last buttons. In the interactive mode, WebMacros detects when the user sidetracks from the recorded path, by matching recorded and actual URL, form data, and page structure. The user can resume playback later from the control panel.

2.5 Encapsulating cookie context in macros

HTTP was designed as a stateless protocol; however, the cookie extension allows servers to store state between HTTP requests. What page is retrieved by a user may depend on the current cookie context. When used with the two popular desktop browsers, Netscape Navigator and Internet Explorer (IE), the WebMacros system optionally includes pre-existing user cookies in a macro during recording. At playback, the user can choose whether cookies included with the macro, or the current ones take priority. Encapsulating cookie context in macros allows their remote playback and sharing.

3. PROXY ARCHITECTURE

The WebMacros system is designed as a pure HTTP proxy (Figure 1). It operates by intercepting the HTTP stream between the user's browser and the WWW and modifying the received HTML. The advantages of the proxy architecture include:

- **A lightweight browser is sufficient for recording and replay.**

As Web access is brought to PDAs and other wireless devices, Web clients other than IE and Navigator must be considered. Web clients for this class of devices are likely to be simple and lightweight; they may not have a built-in JVM needed for the recording and playback applet.

- **A proxy does not need "security clearance".**

For the applet to read/write local files and modify pages, the user must confirm its file and browser access privileges. A proxy does not have this limitation, and can be configured site-wide without user intervention.

- **A proxy enables remote use and sharing of Web macros**

If a trusted third-party server hosts the recording and playback proxy, users can access macros they recorded from multiple computers, and can also share macros.

- **A proxy does not depend on the browser for page retrieval.**

Since a proxy does not use the browser to retrieve macro steps, the proxy can display in the browser only the page retrieved on the last step. In an interactive playback mode, a client-based playback implementation cannot easily determine if the browser has completed loading.

The proxy architecture has also certain drawbacks compared to a client-based one. First, a proxy does not have access to HTML dynamically generated in the browser. In the case of JavaScript, a proxy can either do a partial analysis of scripts (in particular, scan scripts for inline object references), or have a full-fledged JavaScript interpreter. Second, the proxy must provide its own HTML parser for rewriting HTML, since it cannot use the parse tree built by the browser. Third, if the proxy is not installed locally, extra HTTP traffic is generated. Finally, a client-based system for recording and playing macros provides better privacy to users, at the cost of making sharing and remote use more difficult. We believe that the advantages of a proxy solution outweigh its drawbacks, and selected it for our implementation.

4. SYSTEM IMPLEMENTATION

HTML rewriting is the fundamental model used in WebMacros, both to detect user actions that must be recorded as macro steps, and to augment pages with interfaces for controlling recording and playback. When a page is retrieved by a proxy, its HTML parse tree is constructed using the WebL [3] HTMLParser class. During recording, all links and form actions are rewritten to special URLs intercepted by the WebMacros proxy. For example,

<a href=http://online.lib.umn.edu/ovidweb/login.html>

is transformed to:

<a href="http://webmacros/do?_action=record&
_URL=http://online.lib.umn.edu/ovidweb/login.html&
_type=link&_domindex=2">

If the user then clicks on this link, the proxy records the _URL parameter, its type (link vs. form or not-on-page URL), and the Document Object Model (DOM) index on page (2nd link on page in this case) as the next macro step, converts the URL to its original form, and passes it to the Web.

Since a proxy observes all HTTP traffic between the user's browser and the Web, it must distinguish user-initiated requests from those for inline objects, such as images and frames. The WebMacros proxy achieves this by appending unique tokens to inline object references in the incoming HTML, and building a hash of these references. An HTTP request not found in the hash is considered user-initiated and is recorded as the next macro step.

During macro playback, the system reads the URL, DOM information and form field values of each step from a relational database. It then generates a one-line WebL script passed to the WebL interpreter for execution. WebMacros uses a set of heuristic rules similar to the one proposed in WebVCR [1] to account for page modifications between recording and playback time, and to ensure correct macro play

5. FUTURE WORK

We have started testing the system with users to evaluate the convenience and speed of the recording and playback interfaces. The undo/redo features for macro demonstration are being implemented. We are working on detecting iteration during macro demonstration, by looking for similarities in links and forms the user accesses. Intelligently merging multiple pages produced by iteration is another challenge we face.

6. REFERENCES

[1] J. Freire, V. Anupam, B. Kumar, D. Lieuwen. Automating Web Navigation with the WebVCR. *Proceedings of the 9th International World Wide Web Conference*, Amsterdam, Netherlands, May 2000.

[2] T. Kistler and H. Marais. WebL - A Programming Language for the Web. *Proceedings of the 7th International World Wide Web Conference*, Brisbane, Australia. April 1998.

[3] A. Safonov, J. Konstan, and J. Carlis. Towards Web Macros: a Model and a Prototype System for Automating Common Tasks on the Web. *Proceedings of the 5th Conference on Human Factors & the Web*, 1999.

[4] A. Safonov, H. Marais, and J. Konstan. Automatically Classifying Web Pages Based on Page Structure. Submitted to ACM Hypertext 2001.