Logging Usage of AJAX Applications With the "UsaProxy" HTTP Proxy

Richard Atterer
Media Informatics Group
University of Munich
Amalienstr. 17
80333 Munich, Germany
richard.atterer@ifi.lmu.de

ABSTRACT

This paper shows how to use the UsaProxy HTTP proxy to perform logging of user activity for AJAX web applications. UsaProxy is a special-purpose HTTP proxy which modifies HTML pages before forwarding them to the client browser. It adds JavaScript code which collects data about mouse movement, clicks, key presses and other types of interaction without affecting the user's browsing experience in any way. Using Gmail as an example for an AJAX application, the paper explains in detail how to prepare for a UsaProxy-based user test and how to interpret the log files generated by the system.

1. INTRODUCTION

Web applications which make heavy use of JavaScript and DOM pose a problem when trying to obtain meaningful logging data: Much of the code of an AJAX application runs on the client side, modifying the appearance of the page and processing user input without contacting the web server. Instead, the server is only contacted in case data needs to be obtained from the server's database or stored back into the database.

This approach offers several advantages for developers and users of AJAX applications: The web application can often respond to user actions much more quickly, and fewer server resources are needed because the server does not need to recreate the entire graphical layout whenever one detail of the layout changes.

However, from the point of view of someone who wants to obtain detailed information about what people are doing on their web pages, an AJAX application presents a challenge: The data recorded by HTTP servers in their log files is not sufficient to get detailed usage information. With AJAX, there is a separation of client-side code for the user interface and server-side code for application data retrieval/storage. This means that the wrong kind of data (the client-server data exchange) is logged in the server's log, and some crucial information is missing, such as timestamps of specific user actions.

Thus, server-side logs are insufficient when trying to obtain logging data for user interaction. This includes the following scenarios:

- Usability tests of AJAX applications – as mentioned before, not all data (like timestamps of some clicks) can be obtained
- Creation of general usage statistics, for example to obtain data like *"Customers which were interested in the currently viewed item were also interested in..."*
- Analysis of individual users' usage patterns, e.g. for self-adapting websites

In this paper, a solution for logging user interaction with AJAX applications is presented. In section 2, the ideas behind the logging system "UsaProxy" [1] are explained. Section 3 gives a detailed account of the different types of log output created by UsaProxy. Next, section 4 explains how to analyse an AJAX application for a UsaProxy-supported usability test, using Google's Gmail application as an example. Finally, section 5 presents some concluding remarks and discusses possibile improvements to UsaProxy.

2. USAPROXY – AN AJAX-BASED APPROACH TO LOGGING

To address the issues that are mentioned in the previous section, further logging data in addition to the server-side logs needs to be obtained. After taking a look at related efforts [1, section 6], it was decided that the best way to log user interaction for AJAX applications is to also use AJAX technology for the logging task. However, this task was made considerably more complex than one would think at first because of various criteria such as the fact that the logging should not be invasive. The following sections outline the different requirements and the resulting design and implementation details.

Non-invasive and flexible logging system No changes should be necessary to the server-side pages, as this would not be possible for many production websites. Furthermore, no installation of additional software should be necessary at the client side – in the case of volunteers who participate in a usability study from home, this would pose too many technical problems or would not be accepted by the volunteer due to security concerns.

Thus, the tracking is implemented in the form of an HTTP proxy. This approach has been employed by other software such as WebQuilt [2], but none of the existing solutions allows detailed logging of AJAX applications without requiring the installation of client-side software.

It should be noted that in contrast to "normal" HTTP

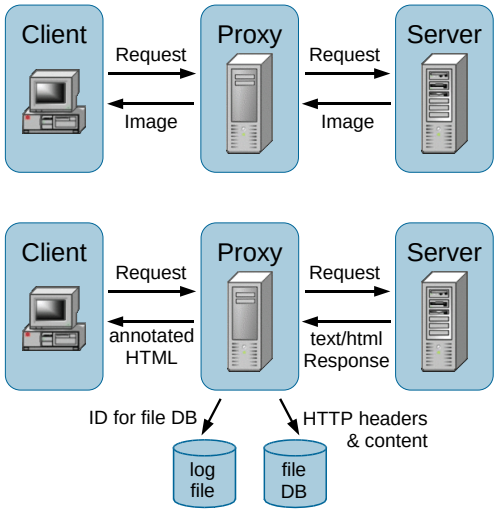

Figure 1: UsaProxy passes on images and other data unmodified (top). HTML content is modified (bottom) by adding special JavaScript. Also, the server response is recorded and identified via a logged ID.

proxies, UsaProxy does not perform any caching of content, it only acts as a filter between the client and the server.

Detailed logging of most interaction on the web pages The system needs to allow recording of as many user actions as possible. This is achieved by executing JavaScript code inside the user's browser. For the current version of the system, the logged information includes the following:

- Loading and unloading of HTML pages. Unloading means that the window for the current page is closed or that a different web page will be loaded and displayed in it shortly.
- Resizing of the browser window
- Giving the input focus to the window or taking it away
- Mouse clicks on the page. UsaProxy records both the page coordinates (from the top left corner of the document) and the HTML element that the user clicked on.
- Mouse movements, in page coordinates
- Scrolling inside the browser window
- Key presses

Logging of mouse movement data using JavaScript and a manually prepared HTML page was demonstrated in [3]. UsaProxy improves this technique, it logs more detailed information and automates the process of adding JavaScript to pages.

In addition to the client-side actions outlined above, UsaProxy also records the browser's HTTP requests and corresponding server HTTP response headers. In those cases where the response has a Content-Type of text/html, the entire response body is also recorded. All log data is written to the file system of the computer that the proxy runs on.

Section 4.2 of [1] describes in detail how the UsaProxy implementation manages to execute logging code on the user's browser and to record the log data. See figure 1 for an overview. The implementation can be regarded as AJAX-based because just like normal AJAX applications, it uses JavaScript and DOM to achieve its purpose. Briefly, the proxy works as follows:

- The client's browser sends a request to the server via UsaProxy. If the corresponding server response is not of type text/html (e.g. if it is an image), the response is simply passed back to the client.
- If the response is text/html, the proxy modifies the page before delivering it to the client, by inserting the following HTML code in the page header:

```
<script src='http://lo.lo/proxyscript.js' type='text/javascript'>
```

- Having downloaded the page, the client requests the above URL. Requests for the special site lo.lo are not forwarded by the proxy, instead they are intercepted. In this case, the proxy sends JavaScript code.
- The proxy's JavaScript code is executed by the browser in the context of the current page, it has all privileges to monitor the user's behaviour.
- When the client-side JavaScript code wants to send log data to the proxy, it does so by making a request to http://lo.lo/img.jpg?*string-to-be-logged*. In this case, the proxy takes notice of the string to be logged. It replies with a "404 not found" response (which is ignored by the client-side JavaScript).

Invisible, AJAX-compatible logging Another requirement for the logging system was that the user's browsing experience should not be altered in any way, but that at the same time, AJAX applications can be monitored with the same accuracy as normal web pages. This goal has been achieved: The proxy's JavaScript runs silently on the user's browser and does not need a lot of outgoing bandwidth to transmit its log data back to the proxy. Furthermore, special care has been taken to ensure that the JavaScript does not interfere in any way with the AJAX application's JavaScript code. Finally, the system is AJAX-compatible because the proxy's JavaScript can follow all modifications to the DOM tree, so it will e.g. always report the correct target element for mouse clicks, even if that element did not exist at the time the page was first loaded into the browser.

3. USAPROXY LOG FILE ANALYSIS

When UsaProxy is used to log user interaction on a web page, the resulting usage data is written to a log file. This section gives a more detailed overview of the different types of log entries. It is based on the list of logged information that was presented in the previous section. The general format of a line in the log is as follows:

client-IP date,time page-url more-details

A typical example for the first part of each line, without the additional event-specific details, is the following:

141.84.120.25 2006-12-31,23:59:59 http://example.org

The URL may also include parameters of a HTTP GET request. In the examples in the rest of this section, only the "more-details" part is shown.

Transmission of text/html content to the client Whenever a web page with text/html content passes through

the proxy, the proxy records the entire HTTP request sent by the browser as well as the server response. This is useful for later analysis of the content that was sent, especially in those cases where HTML was generated dynamically by the server. Non-HTML content is not recorded to reduce the amount of storage needed by the proxy.

In addition to the standard log entry fields, an ID is logged. It uniquely identifies the files to which the request and response have been written:

```
serverdata 15
```

Loading and unloading of HTML pages The client-side JavaScript signals to the proxy when a page has been loaded completely and when a page is left for another page (or the browser window is closed):

```
load width=1394;height=777
unload
```

Unfortunately, the unload event does not work reliably with many browsers, so it will often be missing from the log.

Resizing of the browser window Whenever the user (or JavaScript on a page) resizes the page window, an appropriate log entry is generated:

```
resize width=1050;height=555
```

Changes of the input focus The proxy records whenever the browser window gains focus (i.e. the user clicked inside it, used Alt-Tab to switch to it, or similar) or loses focus (e.g. because the user clicked inside another application's window, switched to another browser tab etc.):

```
focus
blur
```

Mouse clicks For mouse clicks, the tracking code takes note of the coordinates of the click, which is measured in pixels relative to the upper left corner of the HTML page. If the target element can be identified via its name or id, that information is also written to the log. For unnamed elements, the element name is recorded. Even clicks on the scrollbar (which usually mark the start of mouse-based scrolling) can be detected with some browsers. Finally, if the clicked-on element is an anchor, the target URL as well as the link text is logged:

```
click x=57;y=230 target=id:ds_all
click x=144;y=210 target=name:x
click x=512;y=305 target=unknown:DIV
click x=1260;y=3420 target=unknown:scrollbar
click x=292;y=853
  target=link:http://example.org+linktext:Click here
```

Mouse movement All movements of the mouse pointer over the browser window are written to the proxy's log file. However, if every JavaScript onmousemove event were transmitted back to the proxy, this would take up too much bandwidth, so the coordinates are only logged with a rate of about one sample every 300 ms.

```
mousemove x=913;y=674
```

Additionally, mouseover entries provide information on the HTML element over which the pointer currently hovers. The different possibilities are identical to the click entries, for example:

```
mouseover x=57;y=230 target=id:ds_all
```

Scrolling Similar to mouse movements, scrolling coordinates are only recorded with reduced temporal resolution. The logged information is the current vertical offset of the viewport from the document top. For example, a y value of 42 means that the topmost 42 pixels of the page are not currently visible, and 0 means that the page is scrolled to the top.

```
scrolledTo y=42
```

Key presses If the user presses a key, the log output shown below is produced. Keypresses are usually also noticed even if the user invoked a keyboard shortcut of the browser, such as pressing Ctrl-F to start searching for text inside the page. (Unfortunately, the text that is entered in the browser's search window cannot be determined.)

```
keypress key=u
```

4. PREPARING THE USER TEST OF AN AJAX APPLICATION

In the previous sections, the UsaProxy system and its log output were introduced. This section describes the preparations which are necessary to perform a usability test of an AJAX application like Google's Gmail service.

Unfortunately, using the UsaProxy tracking system with Gmail is not straightforward due to the fact that the Gmail login uses HTTPS, which is not currently supported by UsaProxy. A future version of UsaProxy will support HTTPS (and tracking of user actions on encrypted pages). For the moment, one needs to temporarily reconfigure the browser for the login to succeed. As the main Gmail application does not use encrypted HTTP connections, UsaProxy can be re-enabled once the user is logged in.

When preparing for a usability test or similar user action tracking task, the primary effort (apart from setting up the proxy) is to analyse the AJAX application so that the logs created by UsaProxy can later be successfully interpreted.

For example, if one aspect of a usability test is to measure the time taken by a user to accomplish a specific task after entering a website, then one needs to identify the range of possible log entries which mark the start and end of the task.

Similarly, if the aim of the test is to analyse users' navigation patterns when using a web appication, the possible paths (e.g. clicks vs. keyboard shortcuts) need to be linked to specific log entries.

Generally, two approaches are possible:

- Perform a sample session of the user tasks, then look at the logs created by UsaProxy and identify the "obvious" log entries which are of interest.
- Analyse the DOM tree created by the AJAX application and find the nodes which are of interest.

With most applications, the first approach is less complicated and quickly supplies the desired data. However, sometimes the same user action (e.g. a click) may cause different actions (and thus log entries) by the application's JavaScript – in these rare cases, a more in-depth look at the DOM tree may be necessary.

A simple example where the same user action can cause different log entries is when the same button on the page is present on several different layers, and only one layer is

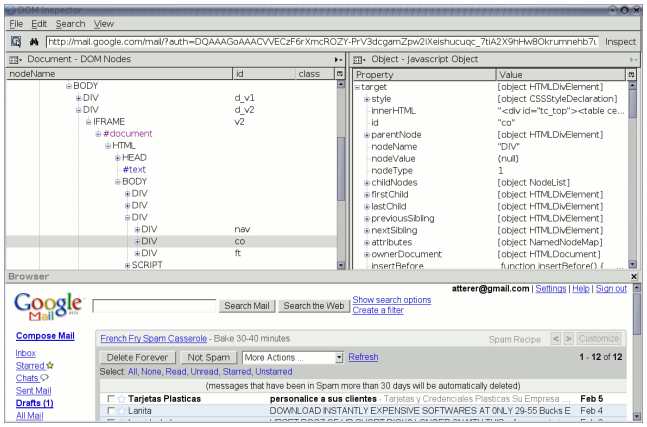

Figure 2: The Firefox DOM inspector tool allows the user to browse through an AJAX application's DOM tree (top-left part of the window) in real time.

displayed at a time. Thus, several copies of the button exist and must be identified.

Performing a sample session When trying out the web application in a sample session to obtain information about those log entries which are of interest for the user test, it is important to use a setup which is as close to the final test setup as possible. For example, the same browser should be used for both, as some AJAX applications may exhibit slightly different behaviour due to browser-dependent implementation details. Furthermore, an attempt should be made to test all possible different ways of completing the task that is given to participants of the user test.

The task of finding the interesting log entries can be made much easier by watching the proxy log in real time as it is being written to disc, e.g. with the tail –f command under Linux/Unix.

For the example AJAX application, Gmail, this approach yields good results. For instance, if we assume that the test participants are set the task "enable POP3 access to the Gmail mailbox", then one can easily determine by looking at the log that a click on the "Settings" link (id:prf_g) is needed, followed by clicks on the "Forwarding and POP" tab (id:pp_d), one of the "enable POP" options (id:bx_pe_2 or bx_pe_3) and the "Save Changes" button (id:ps2).

Analysing the AJAX application's DOM tree For large, feature-rich AJAX applications like Gmail, the structure of the dynamic HTML pages created by the application can be very complex. Still, it may sometimes be necessary to analyse the DOM tree in detail while preparing a user test. However, this can be very difficult: In many cases, it is not possible to look at the HTML pages that are sent by the server because they usually only contain a bare framework of the application's screen layout and all the interesting content is added dynamically by the application code at runtime. An alternative would be to look at the source code and to figure out the names and properties of the HTML elements that one is interested in, but this is equally impractical, as the code of an AJAX application can be very complex. Furthermore, many AJAX applications (including Gmail) come with garbled JavaScript code in which all comments have been removed and all variables given new, meaningless names.

Luckily, tools are available for quick and detailed access to the DOM tree of AJAX applications. Figure 2 shows the Firefox browser's DOM Inspector tool, with Gmail loaded in a browser window inside it. The browser window is fully functional which makes it possible to use the application while analysing it. The document's DOM tree can be browsed in the upper left part of the window, it adapts dynamically in real time to any changes made by the application.

In the figure, the DOM Inspector has been used to find the div element which contains the main page content, i.e. the list of mails together with the controls above and below it. Its id attribute is co.

As an example for a possible use of this information, we assume that the purpose of a user test is to find out how much of the time the mouse pointer hovers over the main content, and how much time is spent in other areas of the page. Using the DOM inspector, we can identify all children of the div element (e.g. by analysing the document subtree which is available via the context menu option "Copy XML"), so that mouseover log entries for the children can be associated with the main content area.

5. CONCLUSION

In this paper, an overview of the UsaProxy logging system from [1] is presented, and the output it generates is explained in detail. Furthermore, it is shown how UsaProxy can be used to conduct user tests of web applications. Google's Gmail service is used as an example to demonstrate the system's use with a real-world, complex AJAX application.

With the example that was used, some limitations of the current UsaProxy implementation become clear. Future work on the proxy will focus on support for encrypted HTTP connections and for even more detailed logging. For example, using the current version, it is sometimes difficult to determine what element has been clicked on (or hovered over) if that element does not have an id attribute.

Acknowledgement This work was funded by the BMBF in the context of the intermedia project.

6. REFERENCES

[1] R. Atterer, M. Wnuk, A. Schmidt: Knowing the User's Every Move – User Activity Tracking for Website Usability Evaluation and Implicit Interaction. In *Proceedings of the 15th International World Wide Web Conference WWW 2006*, Edinburgh, Scotland, May 2006.

[2] J. I. Hong, J. Heer, S. Waterson, J. A. Landay: WebQuilt: A Proxy-based Approach to Remote Web Usability Testing. In *ACM Transactions on Information Systems (TOIS)*, Volume 19, Issue 3 (July 2001), ISSN:1046-8188, pages 263–285

[3] F. Mueller, A. Lockerd: Cheese: Tracking Mouse Movement Activity on Websites, a Tool for User Modeling. In *Proceedings of the Conference on Human Factors in Computing Systems CHI 2001*, extended abstracts on Human factors in computing systems, Seattle, Washington, USA, April 2001