

Developing Web Browser Recording Tools

Using Server-Side Programming Technology

Chris J. Lu Ph.D.
National Library of Medicine
NLM, NIH, Bldg. 38A, Rm. 7N-716, 8600 Rockville Pike
Bethesda, MD 20894, USA
lu@nlm.nih.gov

Anantha Bangalore
National Library of Medicine
NLM, NIH, Bldg. 38A, Rm. 9L-927, 8600 Rockville Pike
Bethesda, MD 20894, USA
bangal@nlm.nih.gov

Tony Tse
National Library of Medicine
NLM, NIH, Bldg. 38A, Rm. 7N-715, 8600 Rockville Pike
Bethesda, MD 20894, USA
tse@nlm.nih.gov


Abstract: It is always a challenge to present Web applications at a facility with no Internet connection. Traditional presentation methods such as transparencies or slides are inadequate for demonstrating dynamic Web applications. Currently, virtual-live demonstrations of Web applications are created with static HTML (Hypertext Markup Language) files. However, preparing such presentations is tedious and requires much manual labor for downloading HTML and other files and modifying them for proper linkages. Demo Server (DS) is a tool that automates this process. Developed using Java Servlets and Apache Web Server on the Sun Solaris platform, DS records various types of browsing activities and saves a set of HTML files for presentation automatically. This paper addresses both system level design concepts and key technical implementation details.

1. Introduction

Even as the Internet becomes popular in research, commerce, and daily life, many places still do not have the infrastructure for connecting to the Internet. Presenting Internet applications in such places becomes challenging. A common work-around is to access a set of static and locally stored HTML files through a Web browser.

A great deal of work is involved in preparing files for such canned demos. Initially, a scenario is sketched, the site is browsed, and all pages are downloaded and saved as HTML files. All hyperlinks and HTML action attributes must be modified and tested in all of the HTML files. Moreover, all image and JavaScript files need to be downloaded for accurate display. Finally, all URLs (Uniform Resource Locator) need to be inspected. The entire process must be repeated each time the scenario is modified. Clearly, such an approach is tedious and not cost-effective.

Demo Server (DS), an Internet browser-recording tool, has been developed to automate the process of creating interactive demonstrations of Internet applications without a live connection to the Internet. DS uses server-side programming (Java Servlets), is a Web based browser independent tool.

At the beginning of a session to create a presentation, DS uploads the page at the URL provided by the user. DS then records all browsing activities, uploading information from each page. Users may stop and download their recording at any time by clicking on the appropriate links. Each HTML file is named as tml [1], where P0.html is the starting page. To begin the presentation, users simply load P0.html in their Web browser.

2. System Architecture and Components

Three major components are implemented in DS (Figure 1): a GUI (Graphic User Interface), a proxy server, and backend modification functions (i.e., parsing functions).

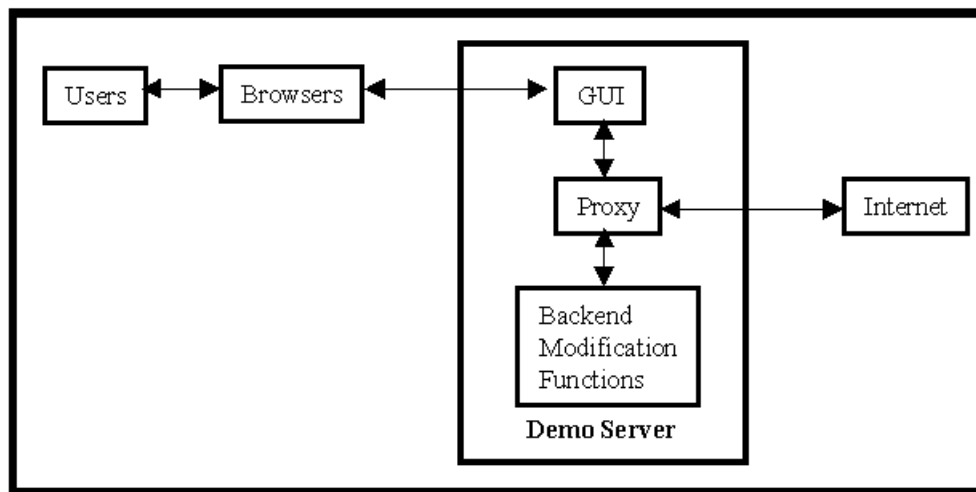


Figure 1: System Architecture and Components of Demo Server.

Users interact with DS through HTML forms via Web browsers. The GUI component provides three major functions. First, it provides a starting point for users to submit the URL and begin a recording. Second, a recording screen displays the Web site that users are browsing and provides a stop button. Third, a final page allows users to save and download recorded files.

The proxy server is the core component linking the GUI component and backend functions with the Internet. As a user starts a recording, the proxy server reads in the URL given by the user from the GUI and then establishes a connection to retrieve contents of the Web site. The proxy then examines the retrieved contents for images or non-embedded JavaScript in the page, automatically downloading the required files from the site. It also keeps track of the current page number during recording, reads and writes HTML content from and to the DS server hard drive, and compresses files for users to download.

The backend component is essentially a parser that searches for and replaces certain strings in the HTML files. In other words, it looks for specific HTML tags and changes the tag's attributes. Three major functions, `modify-1()`, `modify-2()`, and `getImageList()`, are implemented in the DS backend component as described below.

[1] x starts from 0 and increments by one for each page recorded. For example, if four web pages are recorded, there will be the following four HTML files: P0.html, P1.html, P2.html, and P3.html.

`modify-1()` is called when the proxy server gets Web site content from the Internet and displays it to users in the GUI component. Its main purpose is to modify hyperlinks and forms attributes so that retrieved page will access the DS proxy when users click a hyperlink or submit a form. In addition, source attributes of images and non-embedded JavaScripts are modified by this function. Please refer to Section 4 for technical details.

`modify-2()` is called at the end of the recording session and changes the `href` and `action` attributes of hyperlinks and HTML forms in all saved pages to reference itself for all links and forms not clicked or the next page for those that are clicked/submitted [2]. Furthermore, `modify-2()` changes URLs of all images and non-embedded JavaScript files to reference the appropriate directory.

Another function in the backend component is `getImageList()`, which retrieves `src` attributes from `image` and `script` tags in all HTML files and saves them into a list. DS calls this function to download images and JavaScript files from the Internet after `modify-1()` is done.

3. Software Logical Procedures

There are two procedures for operating DS: start and stop recording. Users start recording their Web browsing activities by submitting the starting URL. DS displays the content of the starting page in a separate frame in the lower half of the browser (see Figure 2). However, all subsequent requests go through the proxy component. Users then move to a new page by clicking on a hyperlink or submitting a form while browsing the Web in the lower half of the GUI component. DS repeats the process for every new page (i.e., downloads files from the Internet and modifies them). When users click on the stop button, all modified pages are downloaded from DS to their local computers. The logical steps are summarized below. See Figure 3 for a schematic diagram of the start-recording process.

- **Logical steps for start-recording:**
 1. Users type in the first URL to start the recording.
 2. DS sends out a request to the first Web page. At the same time, DS keeps track of all requests for final modifications.
 3. DS downloads the content of a designated Web site, including images and non-embedded JavaScript files from the Internet.
 4. DS calls `modify-1()` to process the HTML content.
 5. DS saves the modified file to its hard drive. Each downloaded file is named Px.html where x is the current page number.
 6. DS displays the modified HTML file in the browser.
 7. Users move to a new page by clicking a hyperlink or submitting a form on the current page.
Repeat steps 2 to 7.

The logical steps are summarized below and a schematic diagram of stop recording is shown in Figure

4.

- **Logical steps for stop-recording:**
 1. Users send a stop command to DS by clicking on a stop link.
 2. DS calls `modify-2()` to perform the final modification on all saved HTML files.
 3. `modify-2()` reads files from the DS hard drive and modifies them.
 4. DS saves all modified files and compresses (tar) the entire directory.
 5. The compressed files are downloaded to a user's computer upon the user's request.

[2] The proxy server component keeps track of URLs of clicked hyperlinks and submitted forms in all pages. At the end of recording, this information is passed back to function `modify-2()` for final modification.

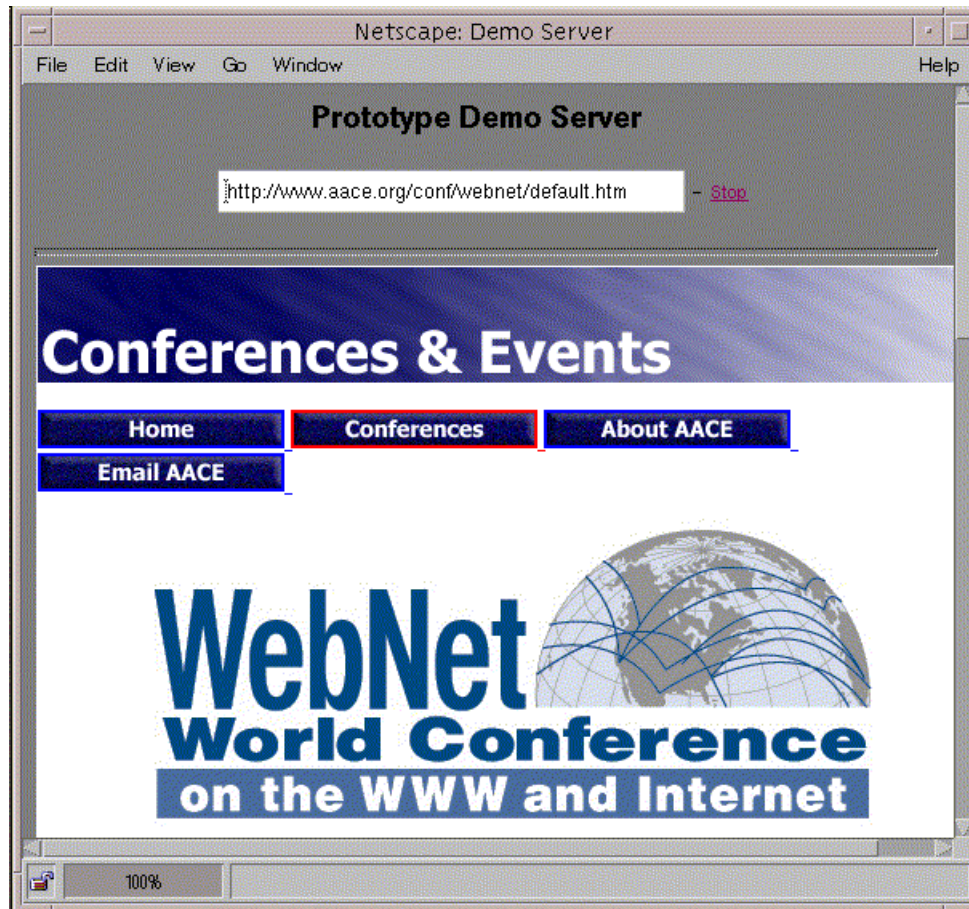


Figure 2: An illustration diagram for Demo Server.

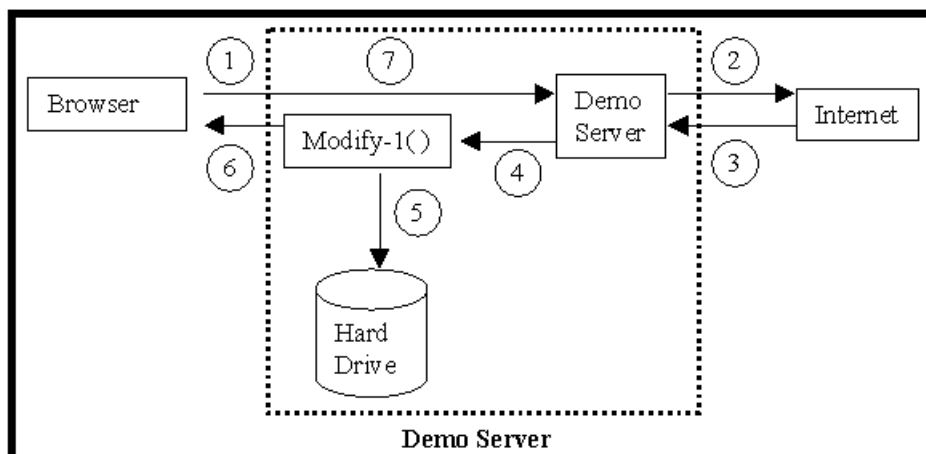


Figure 3: Schematic diagram for start-recording procedures.

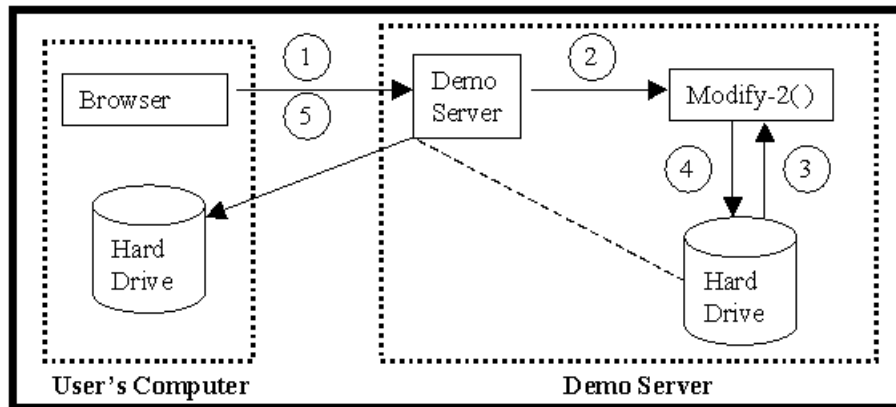


Figure 4: Schematic diagram for stop-recording procedures.

4. Examples and Technical Details

A server-side programming technique, Java Servlets, is used for implementing DS. The Java classes, `URLConnection` and `URL`, are used to get contents from a given URL and download image and JavaScript files. After the first page is displayed, users move from one page to another by either clicking on a hyperlink or submitting a form request. Thus, the key design issues of this tool are handling 1) image and JavaScript files; 2) hyperlinks, and 3) HTML forms. In examples given below, “`dsUrl`” and “`realUrl`” represent the URL of the DS and that of Web site destinations on the Internet, respectively.

- **Image and JavaScript files:**

Images and JavaScript files are handled similarly. Thus, we use image files as an example to illustrate the process. An HTML page that includes an image will have the tag `<image src="http://realUrl/image.gif">`. DS calls function `modify-1()` to change it to `<image src="http://dsUrl/image.gif">` and then displays the page. Users are therefore able to see the image in the DS-modified Web page. After users stop the recording, DS calls `modify-2()` which changes this tag to `<image src="image.gif">`. In this implementation, all image files along with HTML files are stored in the same directory for users to download and replay.

- **Hyperlinks:**

As discussed in Section 2, all hyperlinks on a page need to be modified before DS can display the page since all linking activities go through the DS proxy component during recording. For example, if the current page number is 2 [3], the tag `` is changed to `` by `modify-1()`. With this modification, the DS proxy component is able to parse the content of this hyperlink and retrieve the current page number and the URL of destination page when users click on this link. A final modification is performed by the `modify-2()` function after users stop the recording: The hyperlink becomes `` if this link is clicked and `` if it is not clicked [4]. Thus, users may move from page 2 to page 3 by clicking on this link during replay.

[3] DS keeps track of the page number through the Java Servlets Session function.

[4] If the hyperlink is not clicked during the recording, the link points to itself. This implementation provides a nice feature for users (e.g., incases where a presenter clicks a wrong link by mistake during the presentation).

- **HTML Forms:**

HTML forms are the most complicated cases that DS must handle. For the same reasons as hyperlinks, forms need to be modified before DS displays it to users. For example, if the current page is 2,

```
<form action="http://realUrl" method="POST"> is changed by modify-1() to  
  <form action="http://dsUrl" method="GET">  
    <input type="hidden" name="realUrl" value="http://realUrl">  
    <input type="hidden" name="Page" value="2">  
    <input type="hidden" name="isPost" value="yes">
```

With the above modifications, DS is able to retrieve information from the destination Web site from the Internet, a page number, and a boolean value of isPost [5]. DS reformulates the query and sends a new request to the URL of the Web site while users submit this form to DS. Finally, DS modifies this form again by calling modify-2() after users stop the recording. The final modifications is

```
<form action="p3.html" method="GET"> if this form is submitted and  
<form action="p2.html" method="GET"> if it is not submitted.
```

5. Summary and Future Work

DS is a powerful, cost effective, and easy-to-use tool for users to record and replay Web browsing activities during interactive presentations of Internet applications where no connection are available. It is easy to use and implement. However, there are some issues that need to be addressed to enhance the tool.

DS was originally developed for NLM to present a specific Web site as an interactive “canned” presentation (“http://clinicaltrials.gov”). All HTML files at this site are automatically generated by computer and the syntax is thus very consistent. Realistically, many Web sites on the Internet have inconsistent and even malformed HTML syntax. Currently, DS cannot handle such Web sites. A more sophisticated parsing algorithm is needed to make DS a more generic tool. Besides, the current design of DS assumes all image and JavaScript files have different file names. In addition, DS is currently designed for a single user. Further design modifications are needed to address these issues and incorporate new features into this tool.

Acknowledgements

This research was supported by National Library of Medicine (NLM) at NIH under task order #G0055-036-04. The authors would wish to appreciate Dr. Alexa McCray for her support on this task. The authors also wish to thank Mr. Nick Ide, Dr. Russell Loane, and Ms. Rebecca Crisafulli for sharing their thoughts and discussion.

[5] A Boolean flag, isPost, is used in DS to reformulates (when the value is true) the query by adding variables, such as Page, realUrl, and isPost, to the original GET request.