# Greenbug: A Hybrid Web-inspector, Debugger and Design Editor for Greenstone

David Bainbridge Sam J. McIntosh David M. Nichols
University of Waikato
Hamilton, New Zealand
{davidb,sjmc,dmn}@cs.waikato.ac.nz

## ABSTRACT

In this paper we present Greenbug: a hybrid web inspector, debugger and design editor developed for use with the open source digital library software Greenstone 3. Inspired by the web development tool Firebug, Greenbug is more tightly coupled with the underlying (digital library) server than that provided by Firebug; for example, Greenbug has a fine-grained knowledge of the connection between the underlying file system and the rendered web content, and also provides the ability to commit any changes made through the web interface back to the underlying file system. Moreover, because web page production in Greenstone 3 is the result of an XSLT processing pipeline, the necessarily well-formed hierarchical XML content can be manipulated into a graphical representation, which can then be manipulated directly through a visual interface supplied by Greenbug. We showcase the interface in use, provide a brief overview of implementation details, and conclude with a discussion on how the approach can be adapted to other XSLT transformation-based content management systems, such as DSpace.

## Categories and Subject Descriptors

H.3.7 [**Information storage & retrieval**]: Digital Libraries

## Keywords

General Purpose Digital libraries, Web Inspector, Graphical Debugger, Design Editor

## 1. INTRODUCTION

Striking a balance between expressive capability and ease of use is a challenging aspect to Digital Library (DL) software design. Interface issues, in particular, are a significant source of problems for those learning about digital libraries [2]. Here we present a new level of abstraction to digital library design, fashioned after the web development tool Firebug,[1] but specifically designed for DL software. We start the paper with a demonstration of its features in the context of Greenstone 3 [4] before giving an overview of the implementation and conclude with a discussion of the wider applicability of the developed technique.

[1] https://getfirebug.com/


*JCDL'13,* July 22–26, 2013, Indianapolis, Indiana, USA.
ACM 978-1-4503-2077-1/13/07.

## 2. WALKTHROUGH

Figure 1 shows a snapshot of Greenbug—the name we have given to our developed prototype—in use. Like its namesake, Firebug, initially the software is not active. The digital librarian navigates around the constructed digital library (viewing the home page, query results, preferences information, *etc.*) looking for any imperfections that need to be corrected—or else refinements they wish to make—in the presentation and structure of the DL. When they encounter a page they wish to change, they switch Greenbug on. Unlike Firebug, this is done by logging into the digital library since changes are saved back to the server, thus affecting all users who subsequently access the DL. Figure 1 shows the situation where the digital librarian has logged in because in browsing by titles, they have noticed a rather glaring error present in every title surrogate where it starts with "Aaabbbccc" (artificially introduced for the purposes of demonstration). At the bottom of the browser window the following buttons appear:

Enable debugging; Select new element; Close editor; Save changes; and Use XML Editor,

where only "Enable debugging" is initially active. Clicking this button shifts the user to a mode where, as they move their mouse cursor over the HTML elements on the page, a red highlighting box snaps to the individual element. Clicking on an HTML element of interest calls up internal details from the digital library as to how the page was formed. The nested sequence of XSLT template rules that have been fired to produce the selected element are shown along the bottom of Figure 1. In the figure the digital librarian has chosen to click on the right-most rule: *documentNode*, which in turn displays a visual representation of the XSLT template that produced the selected HTML element.

From here the user can interact with this visual representation, altering the attribute values they contain and rearranging the nested boxes through direct manipulation. The left-hand side offers new elements that can be dragged into the visual representation. New elements are grouped into three categories: HTML, XSL, and GSF (the latter providing DL-based elements, such as access to metadata). Furthermore, the user can interactively select inner-boxes in the visual view to help focus their attention, and the dis-

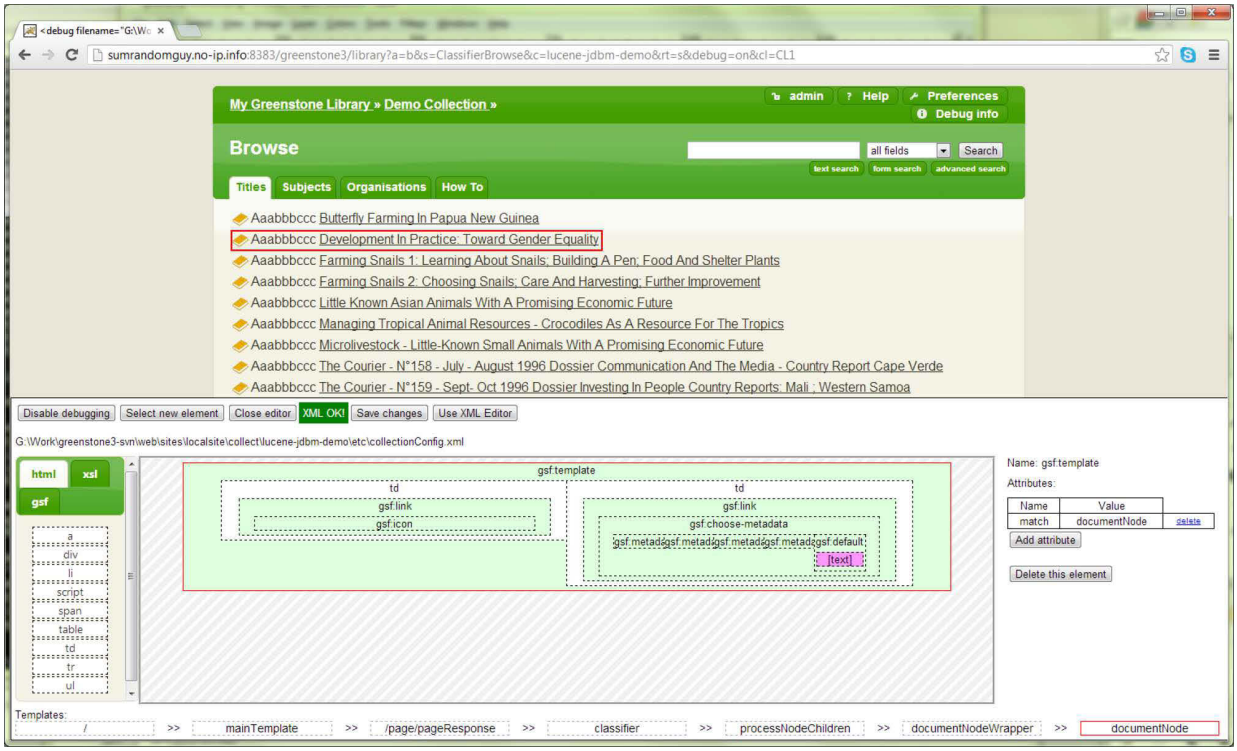


**Figure 1: The Design View to Greenbug.**

play responds by enlarging the area of interest in an animated fashion. Clicking on the outermost-most rectangle reverses the operation. Attributes to the selected element are shown on the right-hand side of the visual display; values can be edited, and new attributes added. In our example, the digital librarian has located the erroneous string "Aaabbbccc", deleted it, and then pressed "Save changes". Greenbug also provides a raw XSLT display mode (not shown) where the user can directly edit XML content in a syntax colour-highlighted editor window.

## 3. DISCUSSION

In terms of implementation, the XSLT processing pipeline to Greenstone 3 [1] was altered to keep track of where the various template rules are located on the file system, factoring in the effect of the inheritance mechanism provided by the <*xsl:import*> element. This information is spliced into the generated HTML as custom <*debug*> elements, relying on the HTML convention that elements that are not understood by the browser are ignored when rendering the page. These debug elements mark the various points within the page where XSLT rules have been applied in generating the page, and attributes within the element record both the name of the template used and its underlying file location.

The second modification we made to the server code was to add an additional (authenticated) service for receiving XML content and saving it to disk. The Greenstone 3 software architecture already has services for saving edited metadata and reconfiguring the server, and the new ability was added to this service group. With these two modifications in place, the end-user interface has been implemented using JavaScript, DOM manipulation, and AJAX calls.

In terms of the general applicability of the approach, there are a variety of XSLT pipeline-based content management systems that would be amenable to being augmented with this form of interactive debugging support, such as DSpace [3] which is based on the Apache Cocoon[2] rendering pipeline. The base functionality required is: user authentication, an instrumented XSLT pipeline, and a URL-based "write file back" and reconfigure service. With these elements in place, the debugging interface (*i.e.*, the JavaScript and AJAX calls) could then be layered on top. In the specific case of DSpace there would be benefit in applying the Greenbug approach to both the Aspects and Themes components of the software design.

## 4. REFERENCES

[1] D. Bainbridge, K. J. Don, G. R. Buchanan, I. H. Witten, S. Jones, M. Jones, and M. I. Barr. Dynamic digital library construction and configuration. *Research and Advanced Technology for Digital Libraries*, pages 1–13, 2004.

[2] D. M. Nichols, D. Bainbridge, and M. B. Twidale. Constructing digital library interfaces. In *Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries*, JCDL '07, pages 331–332, New York, NY, USA, 2007. ACM.

[3] S. Phillips, C. Green, A. Maslov, A. Mikeal, and J. Leggett. Manakin: A new face for DSpace. *D-Lib Magazine*, 13(11/12), November/December 2007.

[4] I. H. Witten and D. Bainbridge. The Greenstone digital library software. *Handbook of Research on Digital Libraries*, pages 61–72, 2009.

[2]http://cocoon.apache.org/