A FRAMEWORK TO DEVELOP META WEB INTERFACES

José Paulo Leal and Hugo Dias
CRACS/INESC-Porto & DCC/FCUP, University of Porto, Portugal
zp@dcc.fc.up.pt c0316035@alunos.dcc.fc.up.pt

ABSTRACT

Web interfaces are used nowadays for virtually every kind of computer application. The proliferation of web interfaces created the need to collect and analyze data on how users interact with them. Many web applications used for this purpose rely on what can be called a *meta web interface*. Meta web interfaces are used for different purposes but they share a set of common features: a web interface based on the subject interface with a second layer interface for collecting data, a central repository for persisting the collected data, and an API for retrieving aggregated data on user interaction. This paper describes Z-Web - a framework for developing meta web interfaces that provides these three features. To create a second layer in the meta web interface a Z-Web server is placed as a proxy between the web client and the subject web server and injects modifications while forwarding HTTP requests. These modifications are typically JavaScript libraries that collect and store data related to user interaction. The framework caches the pages it proxies and provides persistent storage for the collected data. An application interface (API) makes this data available to client application supported by Z-Web. This paper presents an overview of Z-Web, with the general architecture of a web application based on this framework, and describes the design and implementation issues of its main components. Two systems developed with Z-Web are also presented to evaluate the applicability of the framework and its overhead when compared with similar systems.

KEYWORDS

web frameworks, user interaction data collection, usability, web adaption, digital media

1. INTRODUCTION

The Web has evolved from an Internet protocol for sharing static information to a platform for hosting applications and services. From legacy systems to social networks, passing through information systems, office applications and digital news media, virtually all kinds of programs with a user interface are now available on the web. The growing importance of web interfaces increased the need to analyze how users interact with them. There are many research areas where analyzing web interfaces is capital; for instance, in user profiling and usage analysis, in usability testing, in web adaptability research, to name a few.

Web sites need to know their users in detail. The way users interact with an application, their typing speed and the precision with which they point are good indicators of their proficiency (Atterer, R. et al., 2006). Nevertheless, one cannot collect this kind of data using traditional methods, such analyzing HTTP logs. Also, as some pages become larger (e.g. blogs, wikis, news, portals) users spend more time on each page and HTTP servers cannot collected data on which parts of the page receive more user attention.

Usability testing is another area where analyzing web interfaces is important. In this area the concept of implicit interaction is relevant when analyzing human computer interaction (Schmidt et al., 1999). It is important to know how difficult was for the user to complete a task, rather than just knowing if the task was completed. Information systems behind web applications reveal explicit user interaction (what was filled in a form) but give no insight on implicit interaction (how much time it took, which fields where filled immediately, which were corrected before completing the form).

To analyze web pages in these research areas one can use a special kind of web application with a *meta web interface*. Meta web interfaces operate on existing web applications without changing them. Their user interface builds on the interface of the subject web application and extends it with new functionalities. The main advantage of meta web applications is the fact that they are loosely coupled to the web application they analyze. They can operate on existing web applications that do not need to be modified in order to be analyzed. After, or even during, the analysis process the subject web application can continue to be used unmodified.

Meta web applications are typically coupled with subject applications using a variant of a web proxy. Web proxies (Fielding et al, 1999) are used for multiple purposes: to speedup access to commonly requested pages by caching then, to implement or circumvent content filters (e.g. parental filters), to anonymize requests, among many other. When a web proxy is configured in a web browser all HTTP requests are channeled to a specific proxy server. The proxy server relays HTTP requests and responses without changing their content. The variant of web proxy required by meta web applications is different from the classical web proxy in these two points. Firstly, not all HTTP requests are channeled trough the proxy, but only those related to the subject web application. Secondly, subject web application responses are modified to introduce a second layer interface.

The use of a variant of web proxy to inject server side modifications is the main feature of meta web interfaces. This feature is instrumental to modify web pages under analysis without interfering with requests to the subject web server coming from other sources. Other complementary features are the ability to store and aggregate interaction data in a central node and to provide this data further analysis. These features are required by most meta web interfaces, independently from the application area they belong. Thus, we propose Z-Web - a framework to support the development of meta web interfaces providing these common features. The name Z-Web highlights the fact that the main purpose of this framework is to introduce and manage a second layer interface, superimposed over the subject web interface.

The remainder of this paper is organized as follows. Section 2 introduces examples of meta web applications. The following section is an overview to Z-Web and presents the general architecture of a meta web application developed with this framework. The design and implementation issues of the main components are described in detail in the following section. On section 5 we present a couple of meta web applications developed with Z-Web and discuss the overhead it introduces. The last section summarizes the main contributions of this research and highlights future improvements of the proposed framework.

2. RELATED WORK

There are several web applications with user interfaces that can be be described as meta web interfaces. Nevertheless, to the best of our knowledge there are no other framework designed specifically so support the needs of this kind of web applications. The examples available on the literature are mostly of web usability tools. Although not referenced in the literature, there are also some commercial web systems that rely on meta web interfaces.

Websites analyses has been approached in many ways and with different purposes. For example, UsaProxy (Atterer, R. et al., 2006) (Atterer, R. and Shmidth, A, 2007) uses an HTTP proxy to inject JavaScript in the responses from remote servers. With this approach they manage to track user interaction and perform usability tests. WebQuilt (Hong, J. et al., 2001) uses also a proxy based approach to perform usability tests. Although Z-Web can be compared to these examples our objectives are different. The goal of Z-Web is to provide a platform to create meta web interface rather than the actual tests and analyses.

Several commercial web applications rely on the use of proxies that change page content to implement their services. Google Translate provides an automatic translation service for web pages while users navigate on a third party websites. To implement it Google Translate requests pages to the original website, translates texts on-the-fly and return the modified HTML pages to the user. Highlight services also require proxies to modify the content of pages but inject JavaScript libraries to extend the functions of the web pages rather than translating texts. An apt examples of this kind of web application is Roohit.com. One of the case studies in Section 5 is similar to Roohit so this system was used in as a benchmark to evaluate the proxy of Z-Web.

3. OVERVIEW

This section provides an overview of Z-Web, starting with the general architecture of a meta web application developed with this framework and proceeding with an introduction of its main components.

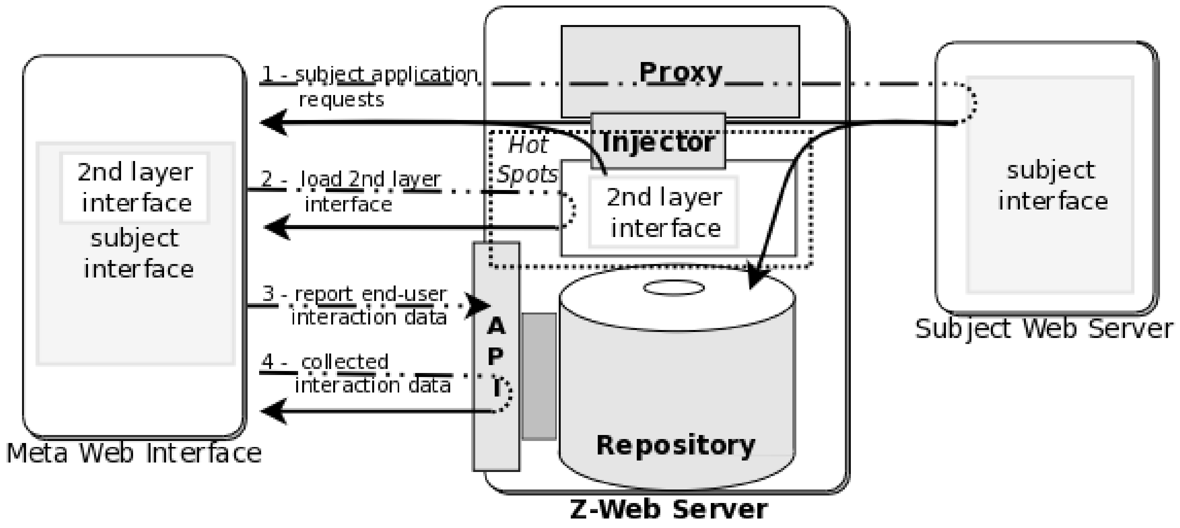


Figure 1 - General architecture of a meta web application based on Z-Web

The general architecture of a meta web application based on Z-Web is represented schematically in Figure 1 and is composed of 3 types of nodes:

Meta Web Interface running on a web client (browser) where the subject web interface is presented with a second layer to collect data on the subject interface;
Z-Web Server capable of mixing requests to the subject web server with requests to create and manage the second layer interface;
Subject Web Server providing the subject web site/application that needs to be analyzed.

The Z-Web sever is the heart of this architecture. It is placed between the meta web interface (browser) and the subject web application server, in a position similar to a classic web proxy. As shown in Figure 1, during its life-cycle the meta web interface makes 4 types of HTTP requests to the Z-Web server: 1) requests to the subject web server are proxied by Z-Web that injects modifications to load extra content (JavaScript libraries); 2) the Meta Web Interface then loads other JavaScript libraries that complement the meta application interface; 3) interaction data collected by JavaScript libraries is reported back to the framework using the API; 4) the same API is also used to access and to aggregate interaction data that is presented in the meta web interface. The Z-Web server that processes these requests is composed of 4 main components:

Proxy - the core element in the framework as it filters the communication with the subject web server; it also articulates with the Repository to cache the pages it proxies and the Injector to change them;
Injector - responsible for introducing modifications on web pages; the default behavior of the injector is to introduce JavaScript libraries that generate and manage the second layer interface; the injected libraries will be able to communicate back with Z-Web using the API;
Repository - a lightweight, document oriented and filesystem based database; it caches web pages and XML files with data related to those pages;
API - exposes the repository to the JavaScript libraries implementing the second layer interface; these libraries may use the API to store and retrieve XML data files associated to web pages, or to collect aggregated data based on these XML files.

Frameworks are composed of non-modifiable parts, also called the frozen spots, and extensible parts, also called hot spots (Pree, W., 1994). The overall architecture of Z-Web and the relationship between its main components are necessarily frozen spots. The two anchor components - the proxy and the repository - are also non-modifiable by user applications. The hot spots of Z-Web are concentrated on the Injector and on the API.

The default behavior of the injector is to modify web pages in two ways: to load an extra JavaScript library and to change relative URLs occurring either in HTML or CSS content. The JavaScript libraries loaded by Z-Web are responsible for creating and managing the second layer interface, thus they are highly dependent on the user meta web application. Changing URLs is necessary to ensure that embed elements, such as images, plugins, stylesheets and other JavaScript libraries, are also processed by the Z-Web proxy. For certain applications the default behavior of the injector may need to be extended to process not only HTML pages and stylesheets but also other text based objects, such as JavaScript libraries.

The Application Interface (API) is another hot spot of Z-Web. It is available to the libraries responsible for the second layer interface, to store and access data associated with web pages. Since web pages and associated data can be seen as resources, we chose the REST architectural model (Fielding, 2002) to structure the API. Data stored and retrieved trough this API is formatted as XML documents (Bray, T., 2006). User applications can define XML Schema (Thompson, H.S., 2001) to validate the data processed trough the API of Z-Web.

4. DESIGN AND IMPLEMENTATION

This section details the design and implementation of the Z-Web server and its main components. As explained in the previous section, the Z-Web server is a specialized HTTP server placed between the subject web server and the web client. For that reason the Z-Web server was developed for a Java servlet container. Tomcat 7, implementing the Java Servlet 3.0 specification, was selected as the server container. It runs on the Java 1.7 virtual machine to benefit from the new input/output API.

4.1 . Proxy and Injector

The proxy is the core element of the framework and its purpose is to process the HTTP request-response cycle of the subject web server. The proxy was implemented using the HTTP Components of the Apache Software Foundation mainly to take advantage of the core HTTP libraries. They were found to be more robust and easy to use than the Java native HTTP API.

The major challenge in this component is to ensure that all the web pages proxied by Z-Web are fully functional. Almost every aspect of the HTTP protocol needs to be processed in a 4 step cycle: 1) the proxy starts by handling the request of the web interface; 2) then it makes a similar request to the subject server; 3) afterwards it processes the subject server response, and; 4) finally the proxy responds to the web interface with a modified version of that response.

The HTTP messages processed by the proxy are composed of a request line, a collection of headers (attribute-value pairs with meta-data), and an optional body (the actual data transferred in the message). Firstly, the requested URL is parsed by the Z-Web server to extract the data it needs to process. These URLs have a common prefix, a data segment and terminate with the URL to proxy. Secondly, the proxy processes the headers. The Content-Type is one of the most important headers because it indicates if the message body must be modified. Other headers related to caching, content compression, and cookies need also to be handled. Cookies were a challenge due to the lack of a single standard for this feature of the HTTP protocol. The proxy supports the most common and well documented specifications, however cookies may not work properly in all web sites. Finally, after processing the headers, the Proxy takes care of the message body using the Injector.

4.3 Repository and Application Interface

The main purpose of the repository is to provide persistent storage to data related to the web pages filtered by proxy, reported via the API. This data is persisted in Z-Web in a document-oriented, file system based, lightweight component that is described in this sub-section. The major decisions when designing the repository were how to encode data, and how to store and retrieve it efficiently. These decisions are interrelated but for sake of clarity this sub-section starts by analyzing how data is encoded, then how it is stored, before detailing the implementation of the repository.

The reason for opting for a document-oriented repository is twofold. Since data is submitted and requested by the client application, its schema cannot be not known when designing the framework. On the other hand, this data is transferred as a document through the API, related to a specific HTML page, and should be preserver unfragmented by the repository. The reasons for opting for a document-oriented repository also support the selection of XML (Bray, T. et al., 2006) as the formalism to encode them. Firstly, with XML we can delegate on client applications the definition of the type of data they need to store, which can be done using a schema language such as XML Schema (Thompson, H.S., 2001). Secondly, XML simplifies the integration with the API since it is a natural encoding for message in web services.

After deciding to encode as XML document the repository data, the following decision is the approach to store and retrieve this data efficiently. Nowadays is becoming a standard practice to avoid the use of traditional relational databases in some domains, notably for storing documents and semi-structured data (Leavitt, N., 2010). This approach to data persistence is known as NoSQL and is mostly used when data is difficult to map into tables, there is not scope to retrieve it using the SQL language, and the burden of a relational system would compromise scalability. Since XML was selected as the formalism to encode documents a possible alternative would be the use a native XML database (Feng, J et al, 2006). Nevertheless, the major advantage of those systems is the possibility of indexing documents and querying them efficiently using languages such as XPath or XQuery. These features are irrelevant for the repository of Z-Web where data is indexed by the URLs of the web pages to which they refer. Hence the decision to merge the HTML cache and data repository in a single component and implement it directly on the filesystem of the Z-Web's server. For that purpose we map URLs in relative path names on the filesystem and save in a single directory both the HTML pages and the data associated with it.

The technologies used to implement the repository were the new Java native Input/Output (NIO 2.0) library and Java Architecture for XML Binding (Ort, E. and Mehta, B., 2003) known as JAXB. NIO improved the repository performance and flexibility when working with the filesystem. JAXB provided a fast and convenient way to bind XML data to Java objects, which facilitated the incorporation of XML documents and the implementation of processing functions of the framework. Additionally, JAXB integrates with Jersey to provide resources in XML and JSON. The repository needs a XSD (XML schema) to be provided in the initial setup to be able to perform validation and to generate at runtime the specific entity classes for the document it will store.

The Application Interface (API) of Z-Web exposes the repository to client application of the framework. Since the repository was designed to contain web pages and associated data in XML documents, this data objects can be seen as resources thus fitting the REST architectural style (Fielding, 2002) chosen to structure the API. This means that the API of Z-Web follows most of the guidelines and constrains of this architectural style to provide access to the data stored by the framework in a simple, scalable and expandable way.

Using one base URL for the web service, clients can interact with the repository without prior knowledge beyond generic understanding of the concept of Hypermedia as the Engine of Application State (HATEOA). Full CRUD operations can be made by the meta web interface or other requests by third party clients.

The Jersey package (Hadley, M et al., 2010) was used for building the API, as it implements JAX-RS (JSR 311), the Java RESTful web services specification. In conjunction, Jersey and JAXB are able to provide representations in different formats and thus Z-Web handles XML, JSON, HTML, plain text and other formats.

5. CASE STUDIES

This section presents two cases studies where the Z-Web framework was used to implement meta web interfaces, both developed to support research projects from third parties. The first case is a web application for research in area of web adaptability and the second case is a web application for research in digital media. The features of this last application are related to those of Rooit, hence we compared the two to evaluate the overhead introduced by Z-Web's proxy on the request-response cycle of the HTTP protocol.

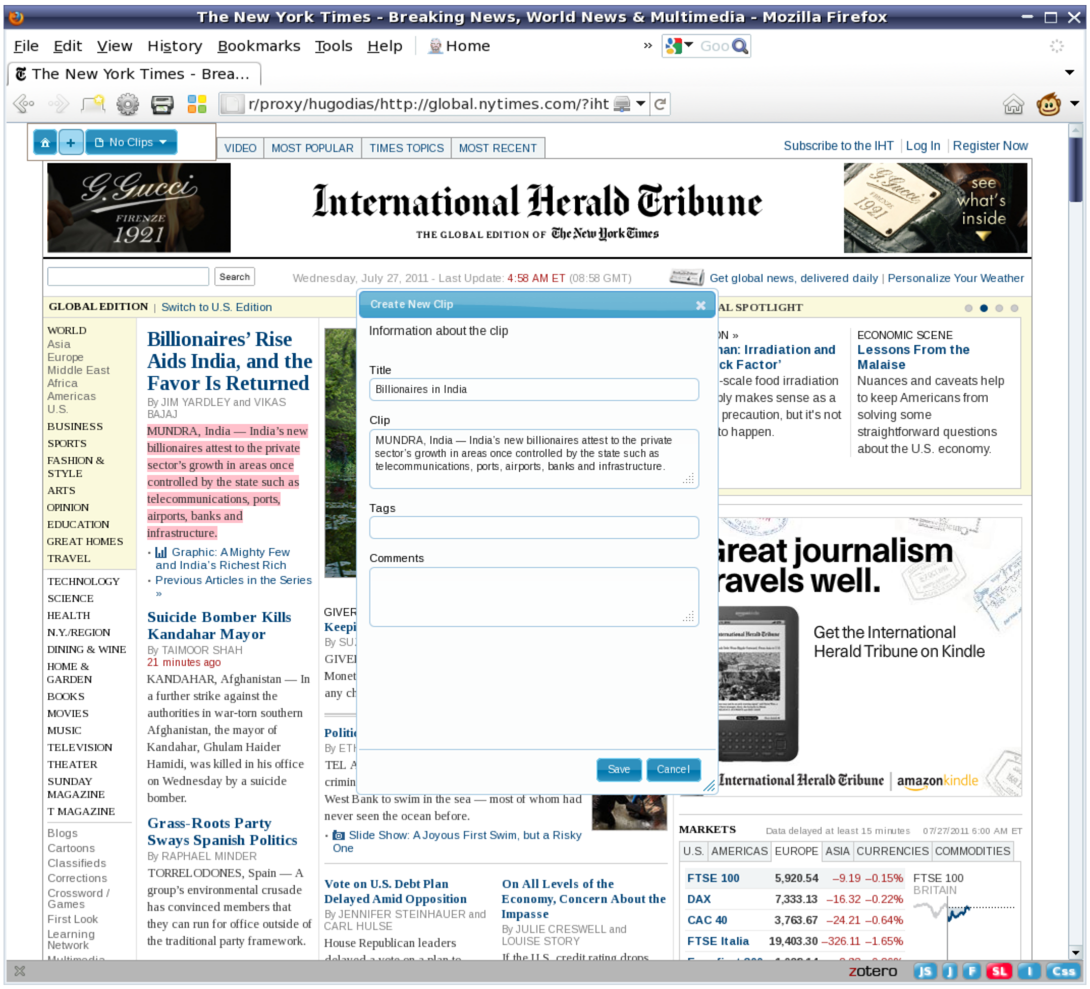
5.1. Web adaptability

The goal of the first application developed with Z-Web was to build a model of user interaction, to detect a moment and a location where the user found an element of interest in the web page. The purpose of such model is to trigger adaptations or recommendations based on the content of the web page that is interesting to a each user. To build such a model it is necessary to collect large amounts of user interaction data (mouse and keyboard events) and process them using data mining techniques. An experiment based on a meta web interface was designed to collect the required data.

The meta web interface for this experiment was composed of two layers. The underlying layer is a web page from a widely used web site (the actual experiment used Facebook and Wikipedia pages). The second layer has a small dialog box with a question. The answer to this question must be found in the underlying web page but should not be easily found by keyword search. The user needs to browse through the page and after finding the correct answer must press the appropriated button in the dialog box. The meta web interface collects the events generated by each user when browsing the web page and stores them using the API. For each page it stores also the exact location of the correct answer (which will vary from browser to browser) and the answer of the user, since only the events from those users that answer correctly are considered valid.

The URL of meta web interface was posted on several social network with a request to participate in the experiment. A total of 734 unique visits where receive in this web application and were selected 299 pages with a correct answer, resulting in a total of validated 228951 events that were then processed.

The implementation of this meta web interface required the codification of a JavaScript library to manage the second layer interface and report collected data using the Z-Web API. This data was encoded as XML and this document type was defined in XML Schema. Both the Javascript libraries and the XML Schema definitions where configured in Z-Web hot spots.


The New York Times - Breaking News, World News & Multimedia - Mozilla Firefox
File Edit View History Bookmarks Tools Help Home
The New York Times - Brea...
r/proxy/hugodias/http://global.nytimes.com/?iht
No Clips
VIDEO MOST POPULAR TIMES TOPICS MOST RECENT
Subscribe to the IHT | Log In | Register Now
G.Gucci FIRENZE 1921
International Herald Tribune
THE GLOBAL EDITION OF The New York Times
see what's inside
Search
GLOBAL EDITION | Switch to U.S. Edition
WORLD Asia Europe Middle East Africa Americas U.S. BUSINESS SPORTS FASHION & STYLE ARTS OPINION EDUCATION GREAT HOMES TRAVEL TECHNOLOGY SCIENCE HEALTH N.Y./REGION DINING & WINE HOME & GARDEN BOOKS MOVIES MUSIC TELEVISION THEATER SUNDAY MAGAZINE T MAGAZINE Blogs Cartoons Classifieds Corrections Crossword / Games First Look Learning Network
Great journalism travels well.
Get the International Herald Tribune on Kindle
International Herald Tribune | amazonkindle
zotero JS J F SL I Css

Wednesday, July 27, 2011 - Last Update: 4:58 AM ET (08:58 GMT)

Get global news, delivered daily | Personalize Your Weather

Billionaires' Rise Aids India, and the Favor Is Returned
By JIM YARDLEY and VIKAS BAJAJ
MUNDRA, India — India's new billionaires attest to the private sector's growth in areas once controlled by the state such as telecommunications, ports, airports, banks and infrastructure.
- Graphic: A Mighty Few and India's Richest Rich
- Previous Articles in the Series »

Suicide Bomber Kills Kandahar Mayor
By TAIMOOR SHAH
21 minutes ago
KANDAHAR, Afghanistan — In a further strike against the authorities in war-torn southern Afghanistan, the mayor of Kandahar, Ghulam Haider Hamidi, was killed in his office on Wednesday by a suicide bomber.

Grass-Roots Party Sways Spanish Politics
By RAPHAEL MINDER
TORRELODONES, Spain — A group's environmental crusade has convinced members that they can run for office outside of the traditional party framework.

GIVER
Keepi
By SU
GIVEI
Monet
any ch

Politi
By ETH
TEL A
crimin
West Bank to swim in the sea — most of whom had never seen the ocean before.
- Slide Show: A Joyous First Swim, but a Risky One

Vote on U.S. Debt Plan Delayed Amid Opposition
By JENNIFER STEINHAUER and CARL HULSE
House Republican leaders

On All Levels of the Economy, Concern About the Impasse
By JULIE CRESWELL and LOUISE STORY

Create New Clip

Information about the clip

Title: Billionaires in India

Clip: MUNDRA, India — India's new billionaires attest to the private sector's growth in areas once controlled by the state such as telecommunications, ports, airports, banks and infrastructure.

Tags:

Comments:

Save Cancel

AL SPOTLIGHT

ON »
han: Irradiation and ck Factor'
-scale food irradiation bly makes sense as a precaution, but it's not to happen.

ECONOMIC SCENE
Lessons From the Malaise
Nuances and caveats help to keep Americans from solving some straightforward questions about the U.S. economy.

MARKETS Data delayed at least 15 minutes 07/27/2011 6:00 AM ET

U.S. | AMERICAS | EUROPE | ASIA | CURRENCIES | COMMODITIES

FTSE 100	5,920.54	–9.19	–0.15%
DAX	7,333.13	–16.32	–0.22%
CAC 40	3,763.67	–24.21	–0.64%
FTSE Italia	19,403.30	–326.11	–1.65%

FTSE 100
BRITAIN

Figure 2 : Screenshot of a meta web interface developed with Z-Web

5.2. Digital Media

A second research project required a meta web interface to collect and classify clips from the web editions of newspapers and other similar information sources. Figure 2 presents a screenshot from the meta web interface developed for that purpose using Z-Web. In this figure the underlying layer is a web page from the international edition of the Herald Tribune. This page was filtered by the Z-Web proxy that added a second layer to manage and collect clips. This layer is composed of a dashboard that is displayed on the upper left corner of the screen. To add a clip the user selects a portion of text and presses the button on the dashboard with a plus sign. The meta web interface then launches a dialog box with the clipped text, where the user can add or change the title, classify this clip with tags and comment it. The dashboard can also be used to manage clips collected previously on the same page. This information is sent to the Z-Web server using the API. The home icon on the dashboard links to a page on the web application where the user can navigate on aggregated data retrieved using the the API of Z-Web.

The steps for implementing this meta web interface were similar to those described on the previous sub-section. Nevertheless, the actual libraries for implementing the second layer interface and the XML schema of the collected data were different and required another setup of the X-Web framework.

It must be acknowledged that the current version of Z-Web still has some issues with the complex web pages, notably with the authentication features of some newspaper web sites. Although the problem has not yet been completely isolated, it is probably due to the difficulty in handling HTTP cookies. However, from the point of view of efficiency the results of this meta web application are very encouraging.

The features of this meta web application are similar to those of Rooit, the highlighting service mentioned in Section 2. This fact makes it a candidate to benchmark the overhead introduced by Z-Web. Table 1 compares the elapsed time (in seconds) for the request-response cycle when fetching HTML pages from several news sites requested directly from their server, and proxied by Z-Web and Rooit. For the proxy times it is presented also in brackets the overhead to the direct server request. The presented times are an average computed from 30 request-response cycles made in sequence and reflect only the time to download the HTML page without its embed elements (figures, flash animations, etc). It can be noticed that Z-Web introduced always a lower overhead when compared to Rooit. In some cases it even outperformed the direct request to the server, although no reasonable explanation could be found for this.

Table 1. Time in seconds and overheads introduced by the proxies of Z-Web and Rooit

Sites	Direct	Z-Web	Rooit
Time magazine	0.12	0.14 (13.39%)	1.38 (91.46%)
Herald Tribune	1.14	1.91 (40.47%)	2.30 (50.53%)
BBC	0.94	0.79 (-19,40%)	2.14 (56.06%)

6. CONCLUSIONS AND FUTURE WORK

This paper presents the design and implementation of Z-Web, a framework to develop applications with meta web interfaces. Starting with an overview of the proposed framework, this paper describes the general architecture of a web application developed with Z-Web, and details the design and implementation of its major components. It concludes with two case studies where Z-Web was successfully used to implement meta web applications.

The first contribution of this research work is the identification and characterization of a class of web interfaces which we named meta web interfaces. This kind of web interface is used for collecting interaction data and analyzing existing web interfaces. Meta web interfaces are typically built in two autonomous layers: an underlying layer coming from the subject web interface, and a superimposed interface layer for collecting information on the previous one. Web applications with this kind of interface are frequently based on a proxy-like server that injects modifications on the web pages of the subject web server while forwarding them. Meta web interfaces require also a central repository to persist the data they collect and an API to retrieve aggregated data.

The major contribution of this paper is the design and implementation of an application framework providing the common features identified in meta web interfaces. The Z-Web framework is a software component to be installed on Java servlet container. It has a number of hot spots where the meta web interface is customized. The default behavior of the framework includes the injection of JavaScript libraries to manage the second layer interface and the actual libraries can be changed for each particular application.

In its current stage the Z-Web framework is already being used in the development of web applications to support research projects in the areas of digital media and web adaptability. These applications demonstrate the adequacy of Z-Web to implement meta web interfaces. They also show that the overhead introduced by Z-Web in the HTTP request-response cycle is negligible; it is actually smaller than that of Rooit, a commercial meta web interface comparable with the digital media application implemented with Z-Web.

The future work in Z-Web will be fixing known issues and writing better documentation. Currently, the main issues are related to certain mechanisms for user authentication, as used in digital media with a subscription for premium services. This issue is probably related to the difficulty in handling cookies due to the lack standardization of this aspect of the HTTP protocol. A better documentation of the framework, in particular of the hot spots for extending the default behavior, is also a fundamental to make the Z-Web framework available to other applications requiring meta web interfaces.

Acknowledgments. This research was partly funded by grant UTA-Est/MAI/0007/2009 (Project Beadcrumbs) from CoLab, the UT Austin | Portugal International Collaboraboratory for Emerging Technologies.

REFERENCES

Atterer, R. and Wnuk, M. and Schmidt, A. (2006) *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 (WWW2006), Edinburgh, Scotland.

Atterer, R. and Schmidt, A (2007), *Tracking the interaction of users with AJAX applications for usability testing*. In Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '07). ACM, New York, NY, USA, 1347-1350. DOI=10.1145/1240624.1240828 http://doi.acm.org/10.1145/1240624.1240828

Bray, T. and Paoli, J. and Sperberg-McQueen C. M. and Maler, E., and Yergeau, F. (2006). *Extensible markup language (xml)*. http://www.w3.org/TR/xml/

Feng, J. and Qian, Q. and Liao, Y. and Li, G. and Ta, N. and Zhou, L. (2006), *Survey of Research on Native XML Databases*, Application Research of Computers, June 2006

Fielding, R.T and Irvine, U et al, (1999) *Hypertext Transfer Protocol -- HTTP/1.1*, IETF Network Working Group, Request for comments 2616, http://tools.ietf.org/html/rfc2616

Fielding, R.T. and Taylor, R.N. (2002-05), *Principled Design of the Modern Web Architecture*, ACM Transactions on Internet Technology (TOIT) (New York: Association for Computing Machinery) 2 (2): pages: 115–150, doi:10.1145/514183.514185, ISSN 1533-5399

Hadley, M. and Pericas-Geertsen, S. and Sandoz, P. (2010), *Exploring hypermedia support in Jersey*, Proceedings of the First International Workshop on RESTful Design, ACM, pp 10--14

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

Leavitt, N., (2010) *Will NoSQL databases live up to their promise?*, Computer, vol. 43, number 2, pp 12--14, 2010, IEEE

Ort, E. and Mehta, B. (2003), XML and Java technologies: Data binding, Part 1: Code generation approaches—JAXB and more, developerWorks—XML or Java Technology. IBM

Pree, W. (1994), *Meta Patterns: A Means for Capturing the Essentials of Reusable Object-Oriented Design*, Proceedings of the 8th European Conference on Object-Oriented Programming (Springer-Verlag): 150–162

Schmidt, A. and Beigl, M. and Gellersen, H. W. (1999): There is more to context than location, Computers & Graphics Journal, Elsevier, Volume 23, No. 6, December 1999, pp 893–902.

Thompson, H.S. and Beech, D. and Maloney, M. and Mendelsohn, N. and others (2001-05), *XML schema part 1: Structures*, http://www. w3. org/TR/2001/REC-xmlschema-2