Personalizing Web Sites for Mobile Devices Using a Graphical User Interface

Leonardo Teixeira Passos[1] and Marco Tulio Valente[2]*

[1] Department of Computer Science, Catholic University of Minas Gerais, Brazil
[2] Department of Computer Science, FUMEC University, Brazil
ltpassos@pucmg.br, mtov@pucminas.br

Abstract. Despite recent advances in wireless and portable hardware technologies, mobile access to the Web is often laborious. For this reason, several solutions have been proposed to customize Web pages to mobile access. In this paper, we describe a wrapper system that targets the personalization of Web pages to mobile devices. A personalization is a subpage of a given web page containing just the information that mobile users would like to access using a PDA.

1 Introduction

Mobile access to the web is often laborious, since most of the web content was not produced for mobile devices. In the last few years, several solutions have been proposed to customize and adapt Web pages to mobile access. In general, these solutions can be disposed on three categories: (i) redesign of web sites for use in mobile devices; (ii) usage of transcoders, i.e., proxies responsible for reformatting web pages, such as Digestor [2] and PowerBrowser [3]; (iii) usage of wrappers, i.e., programs that automatically retrieve and extract particular content from Web documents, such as Lixto [1], WebViews [4] and Smartview [6].

In this paper, we describe a wrapper system that targets the *personalization* of Web pages to mobile computing devices. A personalization is a subpage of a web page that contains just the information a given user would like to access using a PDA. The proposed personalization system – called PWA – meets the following requirements: (i) it supports personalization of standard web pages, despite if they are well-formed or not; (ii) it supports personalizations that are robust to common changes in Web pages; (iii) it supports the creation of personalizations using a GUI, requiring no pattern matching language programming.

The rest of this paper is organized as follows. In Section 2 the architecture of the system is described, along with its algorithms and data structures. Section 3 presents the first results we have obtained in the creation of personalizations and Section 4 concludes the paper.

* This research was supported by FAPEMIG (grant CEX488/02).

2 The PWA System

The PWA system has two basic modules: the personalization definition system and the personalization server. These two modules are described next.

2.1 Personalization Definition System

The personalization definition system is in charge of creating extraction expressions, i.e., expressions that when applied to a web page return the corresponding subpage.

User Interface: In the PWA system the creation of extraction expressions happens in a desktop computer. The following steps are required:

1. The user should accesses the web page he want to personalize using a standard browser.
2. Using the mouse, the user should select the components of the page he wants to access later in his mobile device. The selection process is identical to the one used to select texts in standard editors.
3. The user should copy the selected text to the clipboard.
4. The user should start the personalization definition system and choose the option *New Personalization* in its main menu. From the text saved in the clipboard, the system will generate an extraction expression and will present the personalization associated to this expression to the user.
5. If the user is satisfied with the personalization, he should choose the *Export Personalization* option in the main menu. The personalization will be saved in the personalization server.

This process is fully based on standard browsers and in the built-in *copy-and-paste* facility provided by common window systems.

Data Structures: In order to create an extraction expression, the personalization definition system relies on the following data structures: (i) *list of tags*, i.e., a linear list whose nodes store information about the tags and the text of the target web page; (ii) *buffer*, i.e., an array containing just the raw text of the target web page. (iii) *mapper*, i.e., a function from $Int \rightarrow Int$. A mapping like $i \rightarrow k$ means that the text stored in the i-th position in the buffer comes from the k-th text node in the list of tags.

Figure 1 describes the previous data structures for the fragment of a simple web page. The design of the PWA system was inspired in the MVC architecture [5]. The previous data structures represent the model in the MVC architecture, the browser represents the view and the clipboard the controller.

Extraction Expression: The algorithm used to create the extraction expression has as input value the text T that the user has selected and copied to the clipboard. First, T is matched against the *buffer*, resulting in the range of buffer indexes $[ib, jb]$. If T occurs more than once in the buffer, the intervention of the

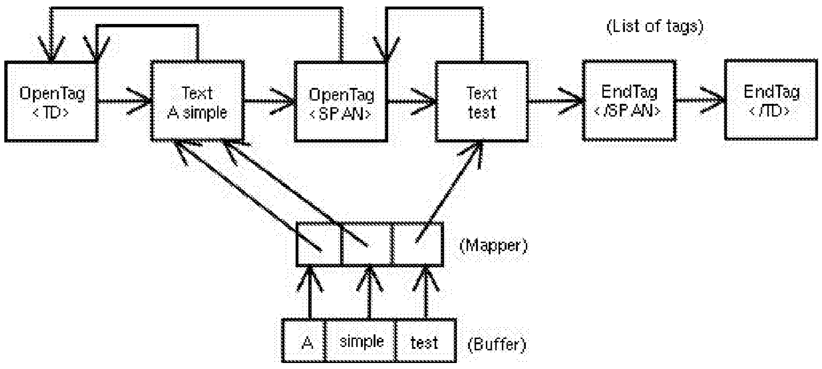

Fig. 1. *List of Tags*, *buffer* and *mapper* structures

user is requested to select the appropriate match. Next, using the *mapper* function, the range $[ib, jb]$ is translated to indexes $[i, j]$ in the *list of tags* containing the nodes where T is stored. The sublist of the list of tags defined by indices $[i, j]$ is called S. The extraction expression is composed by the tuple: (N, U, T, i, j, S), where N is the name given to the expression by the user and U is the URL from the associated web page. This tuple is exported to the personalization server.

2.2 Personalization Server

The PWA server performs the following tasks: (i) it receives requests for personalizations from mobile users; (ii) it retrieves the web page associated to the requested personalization; (iii) using the extraction expression previously informed by the personalization definition system, it extracts the requested personalization from the retrieved web page; (iv) it sends the personalization to the device of the mobile user.

Data Structures: Besides the data structures described on Section 2.1, the personalization server makes use of an extra structure, called *Skeleton*. The skeleton is a copy of the *list of tags* except by the fact that tags that only provide formatting information (such as `<B>`, `<I>` etc), programs (such as `<APPLET>`, `<SCRIPT>`) and images (such as `<IMG>`) are purged. Tags not presented in the skeleton are called *volatile*. Volatile tags are often removed and inserted in web pages. Therefore, expression extractions that consider them can easily break.

Extraction Algorithm: The extraction algorithm has as input value the extraction expression (N, U, T, i, j, S) previously saved in the personalization server by the personalization definition system. The algorithm has the following steps:

1. The page associated to URL U is retrieved and the *buffer*, *mapper* and *list of tags* data structures described on Section 2.1 are created for this page.

2. The text T is matched against the *buffer*. If this match succeeds and the matched indexes are the same indexes $[i, j]$ saved in the extraction expression, the personalization is the sublist S. This happens when the associated web page is static (i.e., its content does not change with the time) and its structure has also not changed since the time the extraction expression was created.
3. If the previous step fails, this means that the page has changed. Then, the idea is to verify if the change is restricted to the content of the personalization, i.e., if the tags (or the structure) of the personalization remains unchanged. Therefore, the sublist S is matched against the full *list of tags* created in the step 1. A sublist S' located in the indexes $[i', j']$ of the list of tags is the returned personalization if it matches S. In case of multiple matched sublists, we chose the one closest to the original position of S, i.e., the match where $|i - i'|$ is minimum.
4. If the previous step fails, this means that both the content and the structure of personalization have changed. The idea is to verify if the changes were restricted to the insertion or removal of volatile tags. First, the *skeleton* data structure is created. Next, the volatile tags of S are removed and a new sublist S' is created. This new sublist is matched against the skeleton. A sublist S'' located in the indexes $[i'', j'']$ of the skeleton is the returned personalization if it matches S'. In case of multiple matches, we chose the one closest to the original position of S, i.e., the match where $|i - i''|$ is minimum.

Table 1. Experimental Results. Column *Success* contains the number of days the personalization was extracted successfully. Column *Changes Inside* contains the number of days the original page has undergone changes inside the area of the personalization. Column *Changes Outside* contains the number of days the original page has undergone changes outside the area of the personalization.

Site	**Success**	**Changes Inside**	**Changes Outside**
Bloomberg (Insight & Commentary)	13	13	13
Bloomberg (Market Snapshot)	13	0	13
Yahoo Sports	13	13	13
Google News	12	13	13

Suppose that P is the personalization returned by the previous algorithm. This personalization is a list containing tags and text. Most of the time, this raw list cannot be directly transmitted to mobile devices, since it does not contain appropriate *context tags*. For example, maybe the first tag of P is a `<TD>`. However, a `<TD>` can never appear without an enclosing `<TR>`, and this last cannot exist without an enclosing `<TABLE>`. Thus, the personalization P is augmented with appropriate context tags before it is sent to the mobile user.

3 Experimental Results

In order to validate the PWA system, we have created personalizations associated to the following sites: `www.bloomberg.com` (two personalizations, comprising the *Market Snapshot* and the *Insight & Commentary* sections of the page), `news.google.com`, (one personalization comprising the *Top Stories* section of the page) and `sports.yahoo.com` (one personalization comprising the top headline section). We have accessed these personalizations during 13 days, in order to verify the robustness of the proposed extraction algorithm. Table 1 summarizes the results we have obtained in this experiment.

As described in Table 1, the personalizations for the Bloomberg and Yahoo Sports web page have remained working for the whole duration of the experiment. The only personalization that has broken during the test was the one associated to the Google News page. This personalization has not produced the expected content in the 13th day of the experiment. The reason was a radical change in the structure of the personalization.

4 Concluding Remarks

We consider that the PWA system presents the following contributions:

– The definition of personalizations by end-users is fully based on a GUI, integrated to a standard browser and to the system clipboard. This GUI is structured according to the well-known MVC model.
– Our first experiments have shown that the proposed extraction algorithm is robust to a considerable types of changes. Particularly, it is fairly robust to changes outside the personalization area. Moreover, its is robust to changes inside the personalization if they affect only volatile tags.

As part of future work, we will focus on the definition of personalizations including different and non-continuous sections of the same web page. Finally, more extensive experiments will be performed.

References

1. R. Baumgartner, S. Flesca, and G. Gottlob. Visual web information extraction with Lixto. In *The VLDB Journal*, pages 119–128, 2001.
2. W. Bickmore and B. Schilit. Digestor: Device-independent access to the world-wide web. In *6th World Wide Web Conference*, 1997.
3. O. Buyukkokten, H. Garcia-Molina, A. Paepcke, and T. Winograd. Power browser: Efficient web browsing for PDAs. In *Conference on Human Factors in Computing Systems*, pages 430–437, 2000.
4. J. Freire, B. Kumar, and D. F. Lieuwen. Webviews: accessing personalized web content and services. In *10th World Wide Web Conference*, pages 576–586, 2001.
5. G. E. Krasner and S. T. Pope. A cookbook for using the model–view–controller user interface paradigm in smalltalk-80. In *Journal of Object Oriented Programming*, pages 26–49, Aug. 1988.
6. N. Milic-Frayling and R. Sommerer. Smartview: Flexible viewing of web page contents. In *11th World Wide Web Conference (poster presentaion)*, 2002.