Personalizing Web Sites for Mobile Devices using a Graphical User Interface

Leonardo Teixeira Passos and Marco Tulio Valente*

Department of Computer Science,
Catholic University of Minas Gerais, 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. Wrappers are one of the most promising technologies addressing this issue. 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. The proposed personalization system meets the following requirements: (i) it supports personalization of standard web pages; (ii) it supports personalizations that are robust to common changes in Web pages; (iii) it supports the creation of personalizations using a graphical user interface. In order to delimitate a personalization, end-users should open the original Web page in a standard browser and click and drag the mouse over the content of the page they are interested to access in mobile devices. In the paper, we describe in details the algorithms and data structures that support the definition and extraction of personalizations. We also discuss our experience in creating personalizations for some web sites.

1 Introduction

Despite recent advances in wireless and portable hardware technologies, mobile access to the Web is often laborious. The main reason is that most of the Web pages were not produced for use in mobile computing devices, i.e., devices with small screens and limited input interfaces. 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 [4]:

– Redesigning web sites for use in mobile computing devices. This is the best solution from the usability point of view and it is commonly employed by high popular web sites, like news portals and electronic stores. However, its main drawbacks is the cost to create and maintain multiple versions of the same content.

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

– Transcoders, i.e., special proxies that dynamically retrieve and reformat documents from the web for use in mobile devices [2, 3]. Transcoders can for example, remove Java scripts and applets, convert HTML to WML, transform colored in black and white images and many other tasks. However, transcoders can not change the overall structure or remove unwanted information from Web documents. Thus, they often generate content that is not adequated to mobile devices.
– Wrappers, i.e., programs that automatically retrieve and extract particular content from Web documents [1, 5]. Given a Web page, wrappers can extract from it only the information that end users consider fundamental for exhibition in mobile devices. Thus, wrappers are a powerful solution to *personalize* Web pages accordingly to requirements previously expressed by final users.

The generation of wrappers however presents many challenges [8]. For example, wrappers generation tools should create wrappers that are accurate and robust, while demanding little effort from users. In this paper, we describe a wrapper system called PWA that targets the *personalization* of Web pages to mobile computing devices. Given a standard web page, a personalization is a subpage of this page created by end-users containing just the information they 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 the specification of robust personalizations, i.e., personalizations that can handle common changes in the source Web page; (iii) it supports the creation of personalizations using a graphical user interface, i.e., end-users do not need to master a pattern matching language, such as XPath [11]. In order to delimitate a personalization, PWA users should open its associated Web page in a standard browser and click and drag the mouse over the content of the page they are interested to access in mobile devices. For this reason, the PWA system follows a so-called *what you select is what you get* strategy to generate personalizations.

The remaining of this paper is organized as follows. In Section 2 we describe the architecture of the proposed personalization system, including the algorithms and data structures used to create and to extract the content of personalizations. Section 3 presents the first results we have obtained in the creation of personalizations for some sites. Section 4 briefly discusses related work and Section 5 concludes the paper.

2 The PWA System

The PWA system has two basic modules: the personalization definiton system and the personalization server. The personalization definition system runs in desktop computers. It is used by end-users to create personalizations from standard web pages and to export personalizations to personalization servers. These servers run in the fixed network since they are proxies that encapsulate actions to retrieve and to extract personalizations from web pages. The resulting personalizations are transmitted to the devices of mobile users.

The details of the personalization definition system and the personalization server are described next.

2.1 Personalization Definition System

The personalization definiton system is in charge of creating extraction expressions, i.e., expressions that when applied to a web page return the subpage that the users would like to access in mobile devices.

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 access the web page he want to personalize using a standard web 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. The user just need to click and drag the mouse over the part of the web page he is interested in.
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, it should choose the *Export Personalization* option in the main menu. The personalition 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. These are tools and tasks incorporated to the routine of users. Thus, they do not need to dominate complex extraction languagens or time-consuming procedures in order to define personalizations.

Data Structures In order to create an extraction expression, the personalization definition system relies on the following data structures:

- List of tags: a linear list whose nodes store informations about the tags and the text of the target web page. Each node also knows the indice of its enclosing tag, i.e., the most inner tag where the text or the tag is nested in. This list is created by an HTML parser that supports ill-formed constructions in web pages (like missing end tags, overlapping tags etc).
- Buffer: a linear list containg the raw text of the target web page. Images and formatting informations are not represented in the buffer. In other words, the buffer stores just the text from a page that can be selected and copied to the clipboard. Each node of the buffer stores a word from the page.

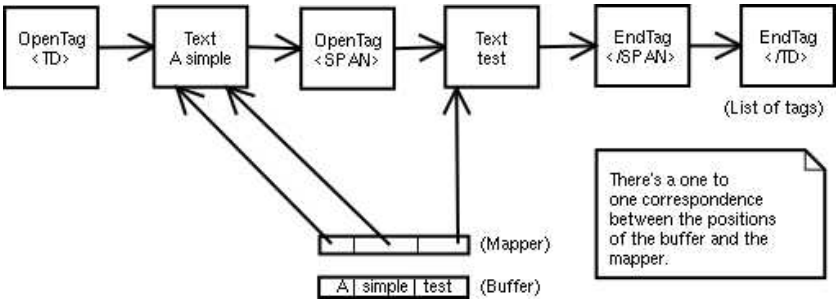

Fig. 1. *List of Tags*, *buffer* and *mapper* structures

– Mapper: 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 tag in the list of tags.

Figure 1 describes the previous data structures for the fragment of a simple web page.

These data structures were inspired in the MVC model (*Model-View-Controller*), originally proposed to support the implementation of GUI systems in the Smalltalk programming language [7]. The MVC model breaks an application into three components: the *model* manages the data manipulated by the application; the *view* manages the graphical output of the application and the *controller* dispatches users requests (mouse operations and key presses) to the model or view components. In the PWA system, the model is represented by the previous data structures. The vision component is represented by the web browser and the controller by the *copy-and-paste* mechanism provided by the window system.

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, the system matches this text against the *buffer*. The result is the range of buffer indexes $[ib, jb]$ that contains text T. If T occurs more than once in the buffer, the intervention of the user is requested to select the appropriate matcth. Next, using the *mapper* function, the range $[ib, jb]$ is translated to indexes $[i, j]$ in the *list of tags* containg 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 uses of the following extra structure:

- 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 `<IMAGE>`) are purged. Tags not present in the skeleton are called *volatile*. Since these tags are commonly inserted and removed from pages, extraction patterns considering them can easily break. Thus, the skeleton is a more robust data structure than the list of tags. It only contains tags providing text that can be macthed in the buffer and tags that define the basic structure of pages (such as `<TABLE>`, `<TD>`, `<TR>`, headers etc).

Extraction Algorithm: The extraction algorithm has as input value an 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 XX are created for this page.
2. The text T is matched against the *buffer*. If this match succeeds and the macthed 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 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 macthes 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 mininum.
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 superfluous 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 macthes 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 mininum.

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, before sending the personalization to mobile devices, appropriate *contexts tags* are inserted in P. The insertion algorithm relies on the pointer to the enclosing tag that is stored on each node of the list of tags.

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.

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

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.

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.

It is worth to mention that all personalizations were associated to very dynamic pages, both on content and on their structure. Despite the second personalization of the Bloomberg site, the remaining personalizations were associated to pages that have undergone changes – both inside and outside the area of the personalization – on each day of the experiment.

4 Related Work

Another personalizaton system for mobile access to the Web is WebViews [5]. Such system is a wrapper that works much like PWA. However, WebViews uses XPath as the extraction language and requires well-formed HTML documents. A pre-processing step using Tidy [10] is suggested to obtain well-formed documents.

Other wrappers, like SmartView [9] and the system proposed in [6], act as proxies that wrap Web pages by the time they are requested by users. SmartView creates logical sections of a document and returns a whole view of it (zoom out) to the user, in a form of an image. To get such image, the proxy server builds a DOM tree of the document of interest and by traversing this tree the logical sections are created. The result is a mapped image representing the HTML page, where each portion corresponds to a logical section. This image is adapted to the dimensions of the user screen. When the user clicks on one of the sections, the proxy returns the corresponding HTML fragment code.

The system proposed in [6] describes a set of techniques to extract relevant content from Web pages without losing their original layout. In order to define what is relevant a specialized user is needed to configure the proxy. Finally, Lixto [1] is a wrapper targeting the discovering of the underlying structure of semi-structured documents.

5 Concluding Remarks

We consider that the design of the PWA system presents the following contributions:

- The definition of personalizations by end-users is fully based on a graphical-user interface. Users should only click and drag the mouse over the text they are interested in and then *copy* it to the clipboard. Besides simple, this interface is based on standard systems, like a Web browser and the clipboard of the window system.
- Since the system is based on a GUI, we decide to structure it following the well-known MVC pattern. In the proposed implementation, the view and controller parts are reused from standard components, such as the browser (view) and the controller (the *copy-and-paste* mechanism provided by the window system). The only component that was really implemented is the viewer, which is represented by the *buffer*, *mapper*, *list of tags* and *skeleton* structures.
- Our first experiments with the system 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 are that 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. In the

current version of the system, this requires the definition of different personalizations. We will also focus on the generation of personalizations encompassing different pages. Finally, we have plans to perform more extensible experiments in order to collect evidences that can improve the robustness of the proposed extraction algorithm.

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. Using wrappers for device independent web access: Opportunities, challenges and limitations. In *WWW Workshop on Mobile Search*, 2002.
5. J. Freire, B. Kumar, and D. F. Lieuwen. Webviews: accessing personalized web content and services. In *10th International World Wide Web Conference*, pages 576–586, 2001.
6. S. Gupta, G. E. Kaiser, D. Neistadt, and P. Grimm:. DOM-based content extraction of HTML documents. In *12th World Wide Web Conference*, pages 207–214, 2003.
7. 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.
8. A. H. F. Laender, B. A. Ribeiro-Neto, A. S. da Silva, and J. S. Teixeira. A brief survey of web data extraction tools. *SIGMOD Record*, 31(2):84–93, 2002.
9. N. Milic-Frayling and R. Sommerer. Smartview: Flexible viewing of web page contents. In *11th World Wide Web Conference (poster presentaion)*, 2002.
10. Tidy Project Page. http://tidy.sourceforge.net/.
11. XPath Language. http://www.w3.org/TR/xpath20.