Adaptive Hypermedia Made Simple Using HTML/XML Style Sheet Selectors

François Bry and Michael Kraus

Institute for Computer Science, University of Munich
http://www.pms.informatik.uni-muenchen.de/

Abstract. A simple extension is proposed for enhancing HTML and XML with adaptation. It consists in using the path selectors of style sheet languages such as CSS and XSLT for expressing content and navigation adaptation. The needed extensions to a path selector language are minimal, a few additional constructs suffice. The processor of the language can be kept almost unchanged, no new algorithms are needed. Furthermore, it is proposed to use XML for expressing user model data like browsing history, browsing environment (such as device, time, etc.), and application data (such as user performances on exercises).

1 Introduction

In existing systems, extending HTML and XML with simple adaptive hypermedia functionalities is done using a combination of cookies, ie client-side user identification, server-side scripting languages like PHP [6], and URIs. This has several drawbacks. Information about the user has to be stored and processed on the server. Due to the nature of the Web's HTTP protocol, this information is limited as compared to the information (possibly) available on the client side: For example, it is not possible to track navigation using the back and forward buttons, navigation in different windows, or navigation on more than one server. This prevents implementing non-trivial adaptive hypermedia systems.

In contrast, the approach outlined in this paper does not suffer from the above-mentioned drawbacks, as it works on the client side. In common adaptive hypermedia systems, the structure of the information, the information itself, and the way of information acquisition together form a user model [2]. This paper does not propose a specific user model, but a framework relying upon HTML and XML that allows a simple implementation of user models. The main advantage of this framework is to make adaptive hypermedia techniques available in the Web context at low cost, ie with minimal changes of the existing standards.

This framework is described in more detail in [3]. It has been proposed to the W3C Device Independence Working Group [4].

2 Browsing Context: A Data Structure for expressing User Models

HTML and XML have no means to express a user model. Therefore, a data structure called *browsing context* is proposed [3], which allows a user model to be stored by the browser, ie on the client side, to be accessed through style sheets, and to be updated through Web applications using scripting languages like Javascript [5]. These features make the data structure "browsing context" convenient for an adaptive presentation of Web pages.

A browsing context consists of three components that can be distinguished according to data acquisition: *browsing history*, *browsing environment* and *application data*. "Browsing history" data are informations about the browsing actions performed by the user in the past such as visiting Web pages, traversing hyperlinks, opening and closing windows, etc. This information is automatically generated by the browser and it is updated each time the user performs a browsing action. "Browsing environment" data are informations about the device (hardware), browser (software), location, time, language, etc. Like browsing history data, this information is automatically generated and updated by the browser. "Application data" are informations specific to the Web application being browsed by the user. In the case of an electronic tutor system, this can be the user performances on exercises, like the numbers of correct answers.

Using style sheets and scripting languages in conjunction with a "browsing context" offers the possibility to easily implement an adaptive hypermedia system cf. [3]. For accessing the "browsing context" with style sheets and scripting languages in a convenient manner, it is preferable to store it in XML format, eg as proposed in [3]. Web browsers store an internal representation of the document currently being displayed, eg as a DOM [1] tree. This document is referred to in the following as *naked document* because it does not contain any browsing context information. In a similar way as this naked document is stored by the browser, a *browsing context document* [3] can also be stored by the browser. Both the "naked document" and the "browsing context document" can be considered as the two parts of one (virtual) *context enriched document* stored within the browser. The "context enriched document" takes over the role of the original "naked document" within the browser, ie style sheets are applied to the "context enriched document" instead of the "naked document", scripting languages have access to the DOM tree of the "context enriched document" instead of the DOM tree of the "naked document", etc. Thus, the "context enriched document" is a virtual document combining a "browsing context" (using which adaptation is expressed) with a standard HTML or XML document. Note that the materialization of this virtual document is not needed.

3 Implementing Adaptation using Style Sheet Selectors

A simple extension to style sheet selectors makes it simple to implement adaptive hypermedia functionalities with HTML and XML. The path expression of a style

sheet selector is not to be matched against the original "naked document" tree, but against the new *context enriched document tree.*

Typical Web style sheet languages like CSS and XSLT have constructs of two kinds: style rules and selectors. Style rules define certain presentation parameters for elements in the document tree (like fonts, colors and margins), and transformations of the document tree (like insertion and sorting of elements). Selectors are path expressions that determine which style rule is applied to which element in the document tree. Matching a path expression of a style sheet selector not against the original "naked document" tree, but against the *context enriched document tree* makes it possible to build path expressions that depend on the content and structure of both, the "naked document" and the "browsing context document". Note that if the path expression of a selector contains no parts referring to a browsing context, the semantics of a style rule remains unchanged. Examples are given in [3].

4 Possible Extensions

Updating Application Data using Scripting Languages. Using style sheet selectors to express content and navigation adaption is not sufficient for modeling certain complex aspects of adaptive hypermedia systems. Still missing is the possibility to store data in the "browsing context", which then could be used by style sheets as a source of adaptation. Scripting languages like Javascript can be used to achieve this. In a similar way as Javascript code contained in Web pages can change the ("naked") document tree, Javascript code contained in Web pages can change the content of a "browsing context"'s application data.

Modeling Locations. There are several different notions of location. (1) Locations can be informations about the country or region where the user is, like Germany or France. This information is available in desktop computer systems and does not change during a browsing session. (2) Locations can be informations about the geographical position of the user, expressed, eg as longitude and latitude. This information is available in mobile devices like cellular phones or PDAs with special positioning equipment, eg a GPS device. Geographical location information can change during a browsing session. (3) Locations can also be informations about *virtual locations* like home, car, office, meeting, etc. Informations about virtual locations can change during a browsing session. "Virtual locations" are represented neither in current computer devices, nor within current Web standards. All of these notions can be represented simultaneously as browsing environment data in a "browsing context".

5 Discussion and Concluding Remarks

The approach outlined here has both, advantages and limitations. First, the approach is quite simple. It introduces a wide range of adaptation features into existing HTML and XML standards at the cost of very limited extensions to

these standards. The extensions to these standards are as follows: (1) Information like browsing history and browsing environment data, most of which is already stored by conventional browsers, is to be stored as a standardized "browsing context" in an internal XML representation like DOM. (2) The style sheet processor(s), eg those of CSS or XSLT, match the selector part of a style rule not against the original "naked document", but against the (virtual) "context enriched document" (consisting of the "naked document" enriched with a browsing context). The style sheet processor must recognize those selector components referring to the "naked document" and those referring to the browsing context. This is conveniently achieved using namespaces.

Apart from these, no further changes are needed, especially, no new algorithms are needed. Only the processing of style sheet rules is extended, the style sheet languages remain otherwise unchanged (because of the use of namespaces cf. [3] section 3). This ensures upward compatibility with already existing style sheets. Also, style sheets that make use of "browsing context" selectors are downwards compatible with non-browsing context enabled browsers. With such browsers the data can be accessed, only the adaptation features are missing. Upwards and downwards compatibilities are essential for extensions to existing Web standards. Thus, the approach proposed in this paper is a conservative extension of the already existing and well-established Web standards.

The approach outlined in this paper is not specific to CSS or XSLT. It relies only on path selectors, which play a central role in Web standards. The same approach can easily be applied to other or future style sheet languages or to other Web standards like XML query languages, as long as they build on path selectors. Note also that this approach is stable against the changes from XPath 1.0 to XPath 2.0, which have introduced a considerably more complex type system, a set of relational operators, and (certain kinds of) variables.

References

1. V. Apparao et al. Document Object Model (DOM) Level 1 Specification Version 1.0. W3C Recommendation, 1998. http://www.w3.org/TR/REC-DOM-Level-1 .
2. P. Brusilovsky. Methods and Techniques of Adaptive Hypermedia. *User Modeling and User-Adapted Interaction*, 6(2-3):87–129, 1996. http://www.contrib.andrew.cmu.edu/ plb/UMUAI.ps .
3. F. Bry and M. Kraus. Adaptive Hypermedia Made Simple Using HTML/XML Style Sheet Selectors. Technical report, Inst. for Computer Science, University of Munich, 2002. Full version of this paper. http://www.pms.informatik.uni-muenchen.de/publikationen/#PMS-FB-2002-1 .
4. F. Bry and M. Kraus. Style Sheets for Context Adaptation. W3C Workshop on Delivery Context, 2002. http://www.pms.informatik.uni-muenchen.de/ publikationen/#PMS-FB-2002-3 .
5. Standard ECMA-262. ECMAScript Language Specification, 1999. ftp://ftp.ecma.ch/ecma-st/Ecma-262.pdf .
6. PHP - Hypertext Preprocessor. http://www.php.net/ .