THE DESIGN AND IMPLEMENTATION OF A WEB-BASED INFORMATION KIOSK

S.HADJIEFTHYMIADES, E.SPANOS, E.TAMBOURIS
Advanced Services Group, Ltd. - 348 Messogion Ave., 15341 Ag.Paraskeui, Athens Greece

The phenomenal growth that the WWW currently experiences, makes it an important candidate for the deployment of interfaces in kiosks intended for the wide public. Although multimedia authoring tools dominated this area in the past, the WWW related technologies have matured and gradually incorporated those elements required for their introduction in kiosk environments. In this paper we discuss certain technical issues related to the design and implementation of WWW based kiosks. We show how we have tackled several problems caused by the peculiarities of a kiosk environment and the need for a very friendly and highly usable interface. (Full paper: http://www.asg.gr/papers/euromed-net.html)

1. Introduction

Nowadays, it is widely agreed that the WWW evolves to the defacto standard for telematic applications in wide area networks. The phenomenal growth of WWW boosted its adoption in corporate intranets for various application types (databases, groupware, etc.). WWW dominance rendered the service a very important candidate for the deployment of multimedia interfaces in kiosks. In general, a kiosk is a publicly accessible terminal to an information system. As the WWW technology attracted the attention of kiosk interface designers-developers, the W3C organised a workshop session during the 3rd International WWW Conference where several technical issues were discussed [1].

In this paper, we show how we have designed and implemented a database oriented kiosk information system by using the WWW technology. We place emphasis on the use of the Javascript and Java languages for resolving kiosk related problems. We discuss the structure of the underlying database which is strongly affected by the "hypertextual" nature of WWW and the need to render the kiosk service maintainable by the non-expert. We also discuss WWW-database connectivity.

The application discussed in this paper has been developed in the context of the **Telematic Applications Programme** (project **RegioNet**, UR 1023), of the EC (DG XIII). The application aims to effectively address the information needs of citizens in two regions of Western Greece. The main focus was on the development of a system that will provide users with information concerning the local authorities and other public services - organisations in the considered regions. The information content of the system encompasses: social security, immigration, education, health services, local transport, etc. The application, which is hosted by a "WWW and database server", is available to citizens through the Internet, direct dial-up connections or a multimedia kiosk.

2. Browser Issues

In all kiosk systems, provision should be taken for returning the application to a starting point when it remains idle for a pre-determined period of time. Although a user initiated interaction with the application and traversed certain nodes of the conceptual graph, he didn't complete what could be generally thought of as a "full session". Such problem is tackled through the use of timers as provided by programming languages like Lingo and OpenScript. A similar feature is not available in standard HTML. Our application is compatible with standard Netscape and Microsoft browsers which can execute JavaScript. Javascript incorporates the SetTimeout primitive that can be used for scheduling the execution of a certain function after a pre-defined time period elapses. To tackle the discussed problem, the Javascript function modifies the location of the current document object thus, forcing the browser to retrieve a new page. The timer value is reset each time the user interacts with the kiosk.

As the application was mainly intended for a kiosk with a low resolution touch-screen device, various GUI elements like combo boxes and hypertext links were avoided. Hyperext links and lists of values (similar to the ones found in combo boxes) were implemented by means of buttons. Such elements were preserved in the version of the application which was available to Internet or dial-up users. The two versions shared the same information content but with different GUI elements. Distinction between the two scenarios was possible due to the WWW-database middleware which was used.

Because of the use of a Web browser in the kiosk, it was necessary to restrict users' access to other applications as well as the operating system (Windows '95). Contemporary browsers offer little support towards this direction. The problem was overcome by creating a series of transparent, topmost - positioned windows. Such windows were placed over the control boxes and the menu bar in

order to filter out the unwanted messages (button clicks, etc.). The relevant program (a satellite application) was implemented in Visual C++ using WINAPI function calls.

Other issues addressed in the design of the application included the presentation of maps which is a very important component in kiosk systems. As it is not possible to pin-point specific positions over an existing map, in the current version of HTML, the use of Java was adopted for this task. Maps of the town/region were stored in popular bitmap formats (e.g. GIF). If some module required the presentation of specific locations over maps, references to the map files were incorporated in the respective database files together with the co-ordinates of the presented locations. When invoked, the Java applet displayed the designated map and pin-pointed the location by interpreting the given co-ordinates. A table containing meta-information on maps was also incorporated in the database.

3. Database Design

In this section, emphasis is given to the relational database which was build in support of the application. In certain cases (e.g. buses, gas stations, hospitals), the designed tables contained references to map files to indicate the relevant locations (e.g bus stops). The table storing map meta-information, apart from typical fields (like map_id and filename), contained the identifiers of four other entries of the same table (recursive relationship). The inclusion of such fields rendered possible the spatial association of the table entries. Maps, when presented to the user, contain a set or guides; their crossing point represents the designated location passed to the applet as dynamic parameter. The map is also surrounded by 4 arrow buttons; one for each direction. The buttons constitute links to the adjacent city maps as they have been specified in the map file entry. This facility assists the user in navigating himself throughout the city map which is fragmented into small files of acceptable resolution. Supplementary zoom-in/out mechanisms are also incorporated in the applet.

In certain application modules a set of organisation charts for the various departments/units were needed. The organisation charts may contain hotspots leading to other organisation charts or to pages of multimedia presentations of the selected department/unit. A top level graphical menu for the selection of a service is constructed on-the-fly. Organisation charts are stored as HTML pages (enriched with Javascript for the setup of hotspots). The pages providing detailed information are constructed on-the-fly and are structured in two frames. The left frame contains a set of pre-defined fields but also other fields whose title and content vary. To cover future requirements, a set (~15) of generic field pairs are provided to the webmaster. Each pair allows for the definition of the title of the field, which will finally appear in the left frame of the HTML page, as well the definition of an anchor. Each detailed service description has an HTML page associated with it. This page contains all the relative narrative and images. The body of the page is broken into segments identified by the aforementioned anchors. When buttons on the left frame are pressed, the system presents (in the right frame) the associated segment of the HTML page. The main rationale behind this mechanism is two-fold: (a) not to limit the multimedia presentation to a static template filled exclusively by the database and (b) to render the module easily maintainable by the non-expert.

4. Interface between the RDBMS and the Web server

As the information that resides in the relational system isn't directly exploitable by the Web server, the use of an interfacing software (middleware) is needed. In the context of the RegioNet project we have made an extensive research on similar tools [2]. For the system discussed in this paper, we have selected Cold Fusion Pro [3]. As an RDBMS we employed Oracle 7.3. Both components were hosted by a Windows NTS 4 system. For its interface to the IIS server, Cold Fusion Pro adopts the ISAPI specification which gives better response times than CGI. Cold Fusion communicates with the RDBMS through 32-bit ODBC drivers. In Cold Fusion, developers build applications by combining standard HTML with a server-side markup language (CFML). Cold Fusion adopts the so-called "template approach". Due to its compliance with ISAPI, Cold Fusion is aware of the IP address of browsers. Thus, it is possible for the application to determine whether a request comes from the kiosk or another user and modify its output accordingly.

References

[1] *Proceedings of the Workshop on W3 Based Online Kiosk Systems*, Third International WWW Conference, Darmstadt (1995).

[2] *Deliverable on System Tools*, Advanced Services Group, Ltd., RegioNet Project, TAP (1996).

[3] *Cold Fusion Pro 2.0, User Guide*, Allaire Corp (1997).