DOCUMENT RESUME

ED 428 697 IR 019 358

AUTHOR Marzo-Lazaro, J. L.; Verdu-Carbo, T.; Fabregat-Gesa, R.
TITLE User Identification and Tracking in an Educational Web Environment.
PUB DATE 1998-06-00
NOTE 7p.; In: ED-MEDIA/ED-TELECOM 98 World Conference on Educational Multimedia and Hypermedia & World Conference on Educational Telecommunications. Proceedings (10th, Freiburg, Germany, June 20-25, 1998); see IR 019 307. For related paper, see IR 019 357.
PUB TYPE Reports - Descriptive (141) -- Speeches/Meeting Papers (150)
EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS Computer Interfaces; Computer Security; Computer Software Development; *Computer System Design; *Computer Uses in Education; Courseware; Database Management Systems; Educational Technology; Foreign Countries; Gateway Systems; Higher Education; Hypermedia; Navigation (Information Systems); *World Wide Web
IDENTIFIERS *Client Server Computing Systems; Computer Users; HTML; Spain

ABSTRACT

This paper describes a solution to the user identification and tracking problem within an educational World Wide Web environment. The paper begins with an overview of the Teaching Support System project at the University of Girona (Spain); the main objective of the project is to create an integrated set of tools for teachers to use to create and publish dynamic and interactive teaching materials that make use of the new possibilities offered by information technologies and the Internet. Functionalities of the Unit Navigation Module (UNM) of this platform that require user tracking and identification are summarized, and previous possible solutions are described. The adopted solution is then presented; the solution uses a Common Gateway Interface (CGI) compliant program running on the Custom Server (CS) machine that processes HTML pages before they are sent to the local (client) machine and inserts user identification within hypertext links. The CS also provides user identification and validation, HTML document customization, database maintenance, translation of database information into reports, control of private working documents, asynchronous and synchronous communications, and other similar capabilities. Three tables present standard log file contents, custom log file contents, and session tracking information. Two figures illustrate standard HTTP architecture and the proposed architecture. Contains 17 references. (DLS)

User Identification And Tracking In An Educational Web Environment[1]

J.L. Marzo-Lázaro*, T. Verdú-Carbó*, R. Fabregat-Gesa*.
* Institut d'Informàtica i Aplicacions. Universitat de Girona.
Avda. Lluis Santaló s/n (17071) Girona (Spain).
Tel.: +34 72 418484 and +34 72 418475. Fax +34 72 418399.
E-mail {marzo,toni,ramon}@eia.udg.es

Abstract : Open Distance Learning is an emerging paradigm where students, teachers, and equipment may be at different geographic locations. The WWW provides an excellent platform for publishing and disseminating a wide variety of curriculum materials. To evaluate the impact of these materials on the teaching and learning process, user identification and tracking is needed. Current HTML clients and HTTP servers are limited in these aspects. In this paper, a solution is described based on an educational web environment. This proposal uses a server program, called Custom Server, that processes HTML pages before they are sent to the client and inserts user identification within hypertext links.

The Custom server takes also the following responsibilities : user identification and validation, HTML-document customisation, database maintenance, translation of database information into reports, control of private working documents, asynchronous and synchronous communications, and other similar capabilities.

1. Introduction

The World Wide Web [1] (WWW or simply "web") is an Internet service that provides an excellent platform for publishing and disseminating a wide variety of curriculum materials. The web is an effective tool for education because of its capacity of presenting multimedia information . Also, the interactive nature of this medium can enhance the teaching and learning process. It can be used as a means to understand complex systems and ideas. It is also a fantastic tool for a constructive approach of this process, in which the educator and the learner are active agents in their respective progress.

Internet communications services such as electronic mail (e-mail), news, Internet Relay Chat (IRC) [2] or videoconferencing tools, facilitate the communication and make the collaboration between students and teachers possible. Individual students can communicate with their teachers and peers without the constraints of having to meet at specific places and times. Students can work on learning materials at own pace and discuss when they have questions, thus sharing experiences with others. They can learn individually but not alone; they can be physically separated but study together through computer networks.

To effectively make use of these materials and evaluate their impact on the teaching and learning processes, user identification and tracking capabilities are needed. Every user must be perfectly identified, and every step he makes must be stored into a database. This database must include for each user which documents have been retrieved, how many times, how much time has been spent within, which have not been visited yet, the responses to proposed tests or exercises, etc. Based on this information the system must generate reports to provide a feedback, to both teachers and students. This information will no doubt help to improve the teaching-learning process.

As we will see further on, current standard Hypertext Transfer Protocol (HTTP) clients [3] (browsers [4]) and servers [5] and HyperText Markup Language (HTML) [6] are limited in user identification and tracking capabilities . The aim of this paper is to provide a possible solution to the user tracking and identification problem within an educational web environment. The presented solution uses a Common Gateway Interface (CGI) [7] program that processes HTML pages before they are actually sent to the client. The CGI program encodes user identification within hypertext links, enabling the server to identify users from modified links passed back from the client. In this way, it provides tracking across a series of HTTP requests while it offers a way to overcome the stateless nature of HTTP sessions without changing the underlying nature of the protocol.

2. The Teaching Support System Project. Objectives

An interdisciplinary group, at the University of Girona, has started the Teaching Support System project [8] and [9]. The main objective of this project is to create an integrated set of tools where teachers can create and publish new dynamic and interactive teaching materials that make good use of all the new possibilities offered by the information technologies and the Internet. The tools are also used by the students to access these materials in a decentralised way

[1] This work has been partially supported by the University of Girona S-UdD97-197 and Spanish Government CICYT : TEL97-1054-C03-03

from anywhere on the Internet, and at the same time by the teachers to keep track of students use. It also improves the communication between students and teachers.

Several objectives of the teaching support system project can be identified. A major goal is to allow teachers to create dynamic and interactive multimedia teaching materials (from now on, "units"), using the new standard information presentation techniques (HTML, Java [10], javascript [11], complemented with Common Gateway Interface programs (CGIs), Server Side Includes (SSI) [12], etc.). The tools are designed to be used by teachers with any level of computing knowledge.

Teachers, students and authorised persons access to the available resources in a decentralised way, speaking both in terms of space and time, and with the benefits of the hypertext concept. The information is structured in a non-linear basis, and therefore each student is capable to access it according to its own learning patterns, needs and initiative. The communication and collaboration between students and teachers is enhanced. The Internet gives us several services that facilitate this communication. The platform must make full use of communications in an integrated way with the materials.

However, this project does not intend to substitute the traditional teaching in classroom, but to complement it. In order to achieve these objectives, our approach has two different main modules. One of them is used by the teachers to make the units, and the other one makes it possible for the students (and other people) to use these units. We call them Units Creation Module (UCM) and Units Navigation Module (UNM) respectively.

The main function of the UCM is the creation of the didactic units by the teachers. But it is important to highlight that we are not only speaking of computer science teachers but all teachers. So the UCM must be easy to use, and it must not require any programming skills nor a lot of new tools to learn. To achieve this last goal, the UCM can make use of materials created using most common standard programs, like word processors, graphics packages, etc. Importing them into predefined templates, the UCM generates the units ready to publish. A complete set of templates is provided to adapt to many types of needs.

The implementation of the UCM module is postponed to a second phase of the project, although some parts are already being developed. In the first phase, we design and implement a first version of the UNM module, and we mainly use already existing standard tools (under a set of basic guidelines) to make the first test units to evaluate and fine tune the UNM. After the UNM is evaluated and validated, the final design and implementation will be made. Then, the UCM module will be designed and implemented.

3. The user tracking and identification problem

The UNM module is the tool used to access the units created using the UCM module. But many of the UCM functionalities require a perfect identification of every single person accessing the system, and must be able to completely track each action this person is performance. For instance, the UNM needs to perfectly know who is accessing which document at any moment, or the answer he gives to any exercise. It would also be interesting to keep some kind of "state" about the user's navigation through the materials, for instance, to enable users to start any new session where they left their previous session. Some of those UNM functionalities that require user tracking and identification are:

Personal Working Space. By clicking a single button, users have access to a personal document with editing capabilities. The objective is to give to the users the "piece of paper and pencil" they certainly need to take notes according as they study the unit. The editor can be either a local one or a Java applet downloaded from the server. In the first case, users are responsible for their documents (name, location, saving it, etc.), while in the second one, the document resides on the server machine, and thus, is automatically loaded and saved when starting or exiting the working space tool.

The Navigation Tree. This tool shows the complete structure of a unit (all their pages, and how are they linked). It makes it possible to "fast jump" to any page of the unit by clicking on the tree. Moreover, this tree informs the user about which pages he has already visited (and how many times, how much time he has spent there, etc.). The Navigation Tree is a useful feature that helps to solve the main hypertext inconvenient: hyperlinked documents are not lineal and therefore it is very easy to "get lost", or forget to visit a whole set of pages only because some link has not been followed.

Complementary Activities, Tests and Exercises. The units can contain complementary activities proposed by the teacher : multiple choice exercises, concept associations, etc. are classical examples. The student gets automatic feedback, being able to immediately know if the answers he gives are correct or not. These activities are created using the UCM, and consist of a set of predefined activities.

Reports. At any moment users can get reports about how the unit is being used. Teachers can ask for a set of reports about the student activity. For instance, teachers will be able to know all the information about any student activity. Also, students can get reports about their own progress. A complete set of activity reports is available.

All these functions, and obviously asynchronous-synchronous communications, have to be implemented with users identification. But current HTTP protocol is very limited with respect to this topic. Therefore, some of the features needed for the UNM are difficult (or impossible), when standard browsers and servers are used. It can be said that the HTTP is an stateless protocol, that is, every request to the server is treated independently, thus a request has no information of previous related requests. In particular, two consecutive accesses made by some students are not related at all from the server point of view. Moreover, a web browser user does not need to identify itself in order to navigate the web, since an HTTP request carries no information about the user, so the server cannot track users.

Therefore, to add identification and tracking capabilities to the basic HTTP client-server scheme is necessary to make all the above functions possible. Moreover, a major design objective is the UNM portability; that is server and platform are independent, and accessible by standard web browsers. We considered some previous possible solutions which are briefly described in the following:

Using the standard log files generated by the server. When a user selects an HTML page link, the browser requests some contents from the HTTP server, (see Figure 1). The server gets and sends the HTML document to the browser. The server keeps several logs for each transfer, but these logs keep only track of content items requested (pages, images, etc.), time, date, the client machine name and IP address. As with most popular operating systems, the use is not identified onto the machine, and the browser does not send any user identification on requests; different users on the same machine generate identical entries into the log file. In this case, whenever different students use the same machine, to keep track of each different student activity is impossible (see Table 1). The server logs only register accesses from the computer, but it is unable to know if they are from a single user or not, and even who is making the requests.

pc1.udg.es - - [22/Oct/1997:18:00:38] "GET /~usd/CS.CGI?2WuJ1uJ1uJdxwqaZSKaCaoM HTTP/1.0" 200 511
pc1.udg.es - - [22/Oct/1997:18:00:38] "GET /~usd/Prog/Users/guest.tmp.html HTTP/1.0" 200 1743
pc1.udg.es - - [22/Oct/1997:18:00:39] "GET /~usd/CS.CGI?504J1uScfwnuA11Z3o1ZWZ7 HTTP/1.0" 200 1834
pc1.udg.es - - [22/Oct/1997:18:15:32] "GET /~bueno/ HTTP/1.0" 200 4477
pc1.udg.es - - [22/Oct/1997:18:15:34] "GET /~bueno/graficos/es0.gif HTTP/1.0" 200 1259
pc1.udg.es - - [22/Oct/1997:18:15:34] "GET /~bueno/graficos/en0.gif HTTP/1.0" 200 1491

Table 1 : standard log file contents

Some entries from a standard log file are showed in the Table 1. Each line represents a request made to the HTTP server. The information shown is relating to the client machine (for instance pc1.udg.es), date, time, the type of request (usually GET), the URL requested (for instance /~usd/Icons/logo.gif). Note that it is impossible to know if requests made from pc1.udg.es (a computers room machine) at 18:00 and 18:15 are from the same user or not. Since standard logs do not save the information that is needed, and they are not a valid solution under our design strategy.

Modify an existent HTTP server in order to adapt it to the tracking and identification needs. Most of standard servers have some Application Programmers Interface (API) that would enable to carry out tracking and identification functions. However, the applications produced for a specific server API are rarely usable by a different server. Moreover, APIs link the application code into the server core, hence, a bug in one API-based application can corrupt other applications or the server, or some malicious application could steal security information from the server. Furthermore, APIs are tied to the internal architecture of a particular server, hence, changes on the server also imply changes on the API, and consequently to the applications which are using the API. In [13] a solution based on APIs is proposed, the client is modified, therefore, similar problems are found. This solution was definitely against our design objectives.

Using Cookies [14]. Netscape proposed Cookies as a mechanism to enable the server side of an HTTP connection to store and retrieve information on the client side; this task is usually carried out by a CGI process. When returning an HTTP object, the server may also send some state information to the client, this state can be stored. The state information includes the range of URLs for which the state is valid. Hence, any future HTTP request made by the client within this range will include the state information, and thus will be passed back to the server. This allows to carry out an identification information within the HTTP requests. Common practice would suggest using these cookies to carry out state and identification information. We discarded it because cookies are not a standard feature, and need non-standard extensions for clients and servers [12]. Moreover, with most common operating systems and current browsers, cookies on a particular client browser are shared by all users who are using the browser, consequently leading to security and identification problems. Cookies can also be easily disabled, copied, modified or erased. Finally, cookies are stored in the client machine, thus the users, who use different computers to log into the system, finish with different state information on each of these computers. For instance, if a user logged in the system using a different client machine would appear in the system as "new" user, since his previous history is ignored. The method to prevent that problem is storing the state information on the server, cookies cannot serve this purpose.

Using standard HTML forms [15]. The forms can contain user identification or any needed state information within hidden variables. The HTML pages that include these forms would be generated dynamically on the server by

CGI programs. When the user pushes the submit button on his browser, these variables are sent back to the server, a CGI could process the information within the hidden variables, and then generate new forms with the new state information into the hidden variables. All served pages should be based around an HTML form, and users should submit them every time they use the application, this is the major limitation of this method. This approach constrains the format of the pages beyond an acceptable limit, as using normal HTML pages would be impossible.

Client-server application using Java. It would also be possible to build a custom client-server application using the Java language. In this case, the user downloads the client from a standard browser; from now on the transfer is carried out using a custom protocol which is designed to support full user identification and tracking. The main drawback is that Java applications are still very slow and resource-consuming at the present.

In [16] a solution is proposed. The major drawback is that an special dedicated client viewer is needed, which must be downloaded and installed by the users before they can start the educational process.

4. The Adopted solution: description of the Custom Server

To overcome the drawbacks of those proposals, while standard browsers and HTTP servers can be used, we propose the utilisation of a CGI compliant program running on the server machine called Custom Server (CS). This approach uses any standard HTML browser running on the local (client) machine and any standard HTTP server running on the remote (server) machine, which support CGIs (see Figure 2). The contents (i.e. the pages including images and Java applets) is written in the HTML format, therefore, any standard editing tool can be used.

The CS is placed between the standard HTTP server and the contents. Consequently, the requested HTML documents are not directly provided by the HTTP server. The information is adequately conformed by the CS by inserting user identification within hypertext links, and redirecting them again to the CS.

In general, the modification of URLs pointing to local HTML files is needed. Note that not all local URLs need to be transformed; most images (and other contents items like Java applets, etc.) do not need to be tracked, hence their URLs are not modified. Therefore, the HTTP server must also be able to access the contents without passing through the CS.

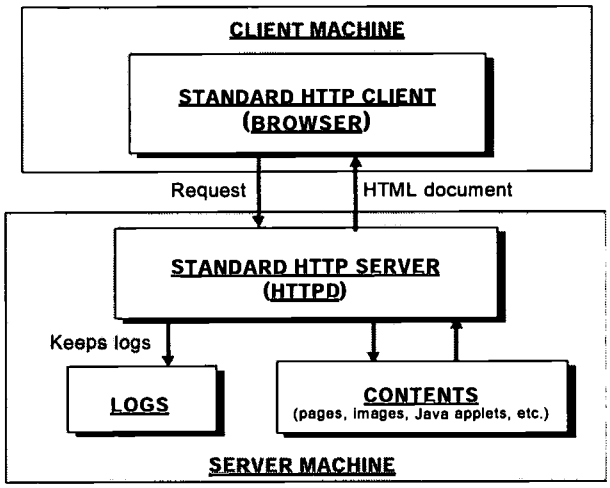

Figure 1 : HTTP Architecture

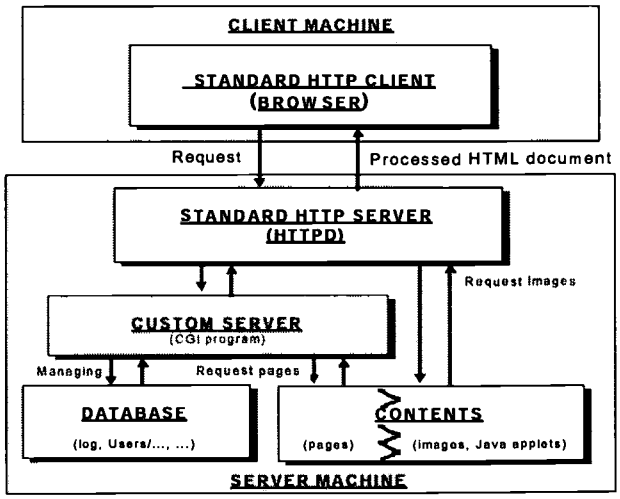

Figure 2 : Proposed Architecture

The contents is stored as HTML pages with "normal" links, and the CS is responsible for the customisation of these pages before they are served according to each user, meanwhile storing information about his navigation steps. Each single page request passes through the CS, thus the CS is able to have a entire control on the accesses mechanism. Any necessary information can be computed, stored and then retrieved at any time.

To store all of this state information about users, the CS maintains a database that overcomes the lacking of default log system which is offered by the Standard HTTP Server. This database includes not only the accessed pages and from which remote machine, but also the information needed to track the user navigation, i.e. which users log into the system, what materials he has accessed, how many times, how much time he spent on it, the responses to proposed tests or exercises, the last pages accessed, etc.

The creation of HTML reports containing the information stored into the database is an important function of the CS. Several reports that users and teachers may need can be easily implemented from this database. Finally, the CS is responsible for another UNM functionality, that is: FAQ accessing and FAQ maintenance, control of working documents, asynchronous and synchronous communications, etc. All of these functions are implemented in the CS.

4.1 The identification process

At the moment of starting the connection, by opening the entry URL [17] using any browser, the user is faced with a login dialogue where, by entering login and the password, he is identified. Without completing this process, the user will not be authorised to use the units on the teaching support system.

The above login procedure has three main purposes : a) to prevent unauthorised users to access the units, b) to be able to identify single users in order to keep track of them when using the units, and customise the context if needed, and c) to be able to distinguish between user categories. The units can behave differently according to the privileges of the user. Some user categories may have access to features others would not.

Related to the third purpose, there are four user categories: Supervisor, Teachers, Students and Guests. The Supervisor acts as the system manager, therefore he has the entire control over the UNM functions. He is also the responsible for assigning login and password to new users, install the units and keep the system running. The teachers have full access to the units, and they can control and supervise their own ones. The students can fully navigate contents of the units and make use of all features that the teacher have prepared for them; but they only can control and supervise their own personal information. Finally, guests have limited access to some units contents, according to the teachers' criteria.

The description of how the identification process is implemented, is explained by an example in the following. First, a user has just entered the login (Enrique) and password (K3db21) on the initial identification page, that is made in a HTML form. By clicking the submit button, the following URL is requested to the remote HTTP server:

http://eia.udg.es/~usd/CS.CGI? Function=ID& User=Enrique& Passwd=K3db21

When the HTTP server gets this request, the Custom Server CGI program (CS.CGI) is invoked. Three parameters (after the "?" and separated by "&" chars) are passed to the CGI on the URL. The first parameter ("Function=ID") informs the Custom Server about an identification request. The first parameter is the function that the Custom Server has to perform. The next parameters are the user name identification (or login) and password that the user just introduced.

4.2 Inserting user identification within hypertext links

The CS must check this login and password in the database. If this is correct, the CS gets the default root page from the contents. Before passing the page to the HTTP server, which will serve it to the browser, the local URLs are customised as is shown in the following. Suppose the starting (root) page contains the following HTML expression:

<A HREF="subject1.html">SUBJECT 1</A>

then the CS explores the HTML source to locate such a local URL, which is transformed to:

<A HREF="http://eia.udg.es/~usd/CS.CGI? Function=PR& User=Enrique& URL=subject1.html> SUBJECT 1</A>

The served page has the modified URL containing the user identification. After some time the user may click on that link, then a new request is passed to the Custom Server. In this case, the value of the function parameter is "PR", that is a page request, the second parameter identifies the user requesting the page, and the third parameter is the name of the page requested.

The CS gets the desired page (subject1.html) from the contents pool, and customises this page in the same manner as before. Subsequently, the CS passes the modified page to the server which actually delivers it to the browser. The CS also reflects on the database that this user is getting this page, and any other needed information.

To simplify this presentation, aspects as parameters encryption to avoid security problems, or questions related to more complex URLs, are omitted. In the current implementation, parameters passed to the CGI are encrypted. URLs contain security information to avoid URL copying and pasting into other browsers that could lead to several problems. The encryption algorithm contains time-out information, hence URLs expire after a reasonable amount of time. It also contains information about the client machine. Consequently, copying URLs from one browser to another on a different machine, the mechanism does not work as desired. Finally, a "request-sequence" information is also used, hence, copying the URL to another browser on the same machine will also fail, preventing the user from "parallel navigation" which could confuse de Custom Server.

4.3 Database management and HTML Report documents

By the user identification, the Custom Server can maintain a database that overcomes the stateless nature of the HTTP protocol. The database consists of a set of log files, similar to the standard HTTP-server log files, but also storing the complementary user information. Mainly, a general log file where users login and logout are stored; but

another particular log file associated with each user is also maintained. In this case, the requested pages and the time spent are saved.

The contents of general log file (Table 2), and user Enrique's log file (Table 3) are described in the following.

Date	Time	Action and Usemame	Client Machine
9/Sep/97	(20:29)	Loginusuario:enrique	(morgan.udg.es).
9/Sep/97	(20:33)	Loginusuario:toni	(atreides.udg.es).
9/Sep/97	(20:42)	Usuario:enrique logout de la cuenta.	

Table 2 : Custom log file contents

Requested HTML-document	time spent in seconds
*Unidad: LES XARXES DE COMUNICACIONS I LA INFORMÀTICA	
teleinfo/index.html	9 (00:00:09)
teleinfo/repercusions/comunicacio.html	227 (00:03:47)
teleinfo/index.html	330 (00:05:30)

Table 3 : One session tracking information.

The Custom Server analyses these files and translates the information into HTML reports. By consulting these reports, both teachers and students may be able to evaluate how the materials are being used, and to improve the teaching-learning process. In [12] a similar solution is proposed for similar objectives. But, since all state variables are embedded into the URL, the state cannot be preserved between different sessions at the application level.

5. CONCLUSIONS AND FUTURE WORK

In this paper, a solution to the user tracking and identification upon a educational web environment is proposed. This approach is implemented into a wide project that needs to keep complete track of users who access to a set of contents published on the net. This approach is relatively easy to implement, it also gives enough flexibility and it is compatible with any standard HTML web browsers and HTTP servers.

This is a work in progress, consequently, some parts are still being implemented. Therefore, some above aspects may change in the future, new functionality may be added, like new reports, according to the suggestions of the actual users. The adopted solution gives sufficient flexibility to adapt the platform to future needs.

References

[1] W3C - The World Wide Web Consortium. *http://www.w3.org*

[2] RFC1459 : Internet Relay Chat Protocol.

[3] Hypertext Transfer Protocol Overview. *http://www.w3.org/pub/WWW/Protocols*

[4] BrowserWatch. *http://browserwatch.iworld.com*

[5] The Netcraft Web Server Survey. *http://www.netcraft.co.uk/Survey*

[6] HyperText Markup Language (HTML) *http://www.w3.org/pub/WWW/MarkUp*

[7] Common Gateway Interface *http://www.w3.org/pub/WWW/CGI/Overview.html*

[8] T. Verdú and R. Fabregat "Uso de las nuevas Tecnologías de la Información e Internet, como complemento de Innovación y Mejora de la Docencia" - Congreso Universitario sobre Innovación Educativa en las Enseñanzas Técnicas Set.1.996. Zaragoza

[9] J.L. Marzo, M. Estebanell, R. Fabregat, F. Ferrés, T. Verdú. "Support Units for University Teaching based on WWW". ED-MEDIA/ED-TELECOM'98. June 1.998. Freiburg.

[10] JavaSoft Home Page. *http://java.sun.com*

[11] JavaScript Guide. *http://home.netscape.com/eng/mozilla/3.0/handbook/javascript/index.html*
JavaScript 1.1 Language Specification *http://home.netscape.com/eng/javascript/index.html*

[12] SSI+ 2.0 Reference. *http://webquest.questar.com/reference/ssi/ssi+20ref.sht*

[13] A. Iyengar. "Dynamic argument embedding : preserving state on the World Wide Web". IEEE Internet Computing. Volumen.1 Number 2. March - April 1.997. pp. 50-56

[14] Persistent Client State. HTTP Cookies. Preliminary Specification. *http://home.netscape.com/newsref/std/cookie_spec.html*

[15] HTML Forms and CGIs. http://www.chin.gc.ca/~training/html/forms.html

[16] B. Ibrahim and S.D. Franklin. "Advanced - Educational Uses of the World Wide Web". WWW'95. Third International WWW Conference - Technology, Tools and Applications April 10-14, 1995, Darmstadt, Germany.

[17] Web Addressing Overview. *http://www.w3.org/pub/WWW/Addressing*

Acknowledges

The authors would like to thank MeritxelEstebanell Minguell and JosefinaFerrés Font, Departament of Pedagogy, The University of Girona, for their many constructive observations and comments. Also thanks to EnriqueMonserrat for his valuable help in the implementation of the Custom Server software and for his many technical contributions.

U.S. Department of Education
Office of Educational Research and Improvement (OERI)
National Library of Education (NLE)
Educational Resources Information Center (ERIC)

NOTICE

REPRODUCTION BASIS

☒ This document is covered by a signed "Reproduction Release (Blanket) form (on file within the ERIC system), encompassing all or classes of documents from its source organization and, therefore, does not require a "Specific Document" Release form.

☐ This document is Federally-funded, or carries its own permission to reproduce, or is otherwise in the public domain and, therefore, may be reproduced by ERIC without a signed Reproduction Release form (either "Specific Document" or "Blanket").