Object Oriented Web Client for Content Management System

Ivan Petković[1], Milena Stanković[2]

Abstract – **Data redundancy is usual on the client side of the contemporary Web applications. Eliminating the redundancy can significantly improve Web site performance and maintenance. For that purpose, we propose a new efficient method which eliminates redundancy on the client side.**

Keywords - **object oriented, client side, content management system**

I. INTRODUCTION

Managing content has always been one of the primary needs for using the computer. At the earlier stages of the Internet evolution, Web sites contained primarily static content. Another problem was lack of separation between presentation and actual content. This lead to the fact that only people who knew the actual code implementation of the Web site could change the content. Now days, as technology advanced, both the types of content and the way people manage and use it have greatly changed. New content types such as continuous multimedia streams have become commonplace due to the constantly improving storage, encoding, and networking technologies. This combination has allowed users to access and share multimedia content in both local and remote area networks with the network itself acting as a huge data repository [1]. Two issues became very important: how to efficiently find and use Web content, and how to produce and make them accessible to users. Solution to the first issue is to design a good Web client interface, and for the second issue is to make a content management system.

In section II we will discuss about the needs for content management systems, and afterwards, in section III, we will describe the Web content lifecycle in. Section IV will explain the architecture of the general-purpose content management system, and section V will be more specific about the client side of the content management system. In section VI we will describe proposed Web client.

[1] Ivan Petković is with the Faculty of Electronic Engineering, Beogradska 14, 18000 Niš, Yugoslavia, E-mail: ivanp@elfak.ni.ac.yu

[2] Milena Stanković is with the Faculty of Electronic Engineering, Beogradska 14, 18000 Niš, Yugoslavia, E-mail: mstankovic@elfak.ni.ac.yu

II. NEEDS FOR CONTENT MANAGEMENT SYSTEMS

A content management system is a system used to manage the content of a web site. It typically consists of two elements: the content management application and the content delivery application. First one allows the authors, to manage the creation, modification, and removal of content from a web site, while the other is focused on the content delivery to the users.

The need for the content management systems (further referred as CMS) evolved for the series of reasons, but we should state the major ones:

- Site visitors have difficulties to find what they need
- Content must be updated often
- Content must be modified by a different people
- Content must be extracted from the various sources (database, XML, other applications, etc.)
- Content must be available to the different types of clients (PDA, WAP, Web of other)

There is also a set of requirements CMS should meet. Two fundamental requirements every CMS should provide are separating the way of presenting contents (layout) from the actual content, and separating the site map ("where it is stored") from content. This enables content redeployment to different locations and devices, with changeable "look and feel" of the client interface. Because a large number of people should be able to use the CMS, it must operate on a large number of heterogeneous systems. It should also be flexible enough to support content management policy (what content should be obtained, how it will be integrated, and under what circumstances should it be discarded).

III. ARCHITECTURE OF THE CMS

There are different opinions on the architecture of the content management system, but we can define fundamental model which can be modified according to the specific needs (Fig. 1). This model is divided into three sections:

- client
- CMS engine
- source

We should state that the presented model is for general-purpose CMS, not only for Web. Web content management

systems are more specialised since their primary focus are Web clients. Clients represent target platforms which receive the content. Since the client platforms can drastically differ (e.g. cell phones, Web pages and PDF documents), content must be specialy adapted for each of them. As we stated earlier, this can acheived by following the "golden rule of CMS" – separation of storage, presentation and content.

Content sources can be very different, and they can also be distributed. Most frequently used type of source is database, which is often used in Web content management systems. Besides database, there are other types of sources: Web services, structured and unstructured documents, and even input from machines.

In order to acheive univesal data interchange, inputs from all the sources must be transformed (adapted) to the standardized form. Nowdays, the most promising solution is using XML as an open standard based on common synax and with infinite semantics.

Our primary focus will be on Web clients and their design and implementation using the proposed object oriented approach.

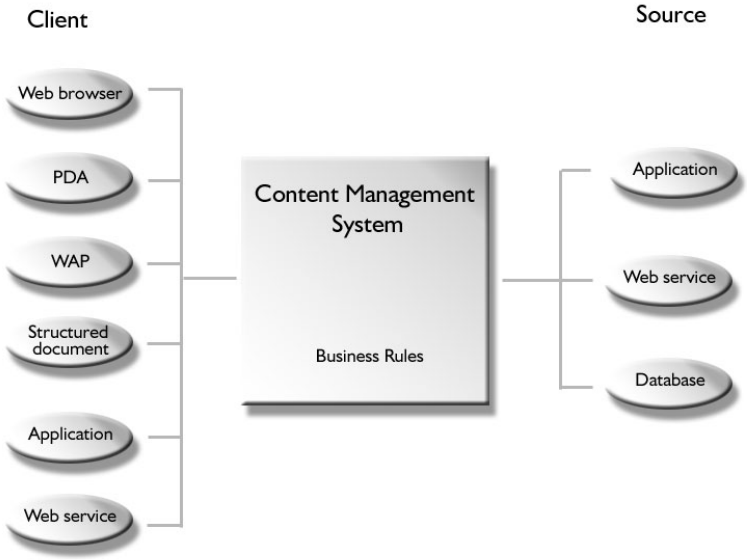

Fig. 1 Architecture of the CMS

IV. THICK WEB CLIENT

There are two Web Client patterns used in Web development:

- Thin, and
- Thick Web Client.

Thin client usually refers to a system that runs on a resource-constrained machine. The client only requires a standard web browser. All of the business logic is executed on the server.

In principal, Thick Web Client pattern include the dynamic of the Thin Web Client pattern plus the ability to execute all or some parts of the business logic on the client. As with the Thin Web Client pattern, all communication between the client and server is done during page requests. The business logic however, can be partially executed on the client with scripts, controls or applets.

Page sent to a client may contain scripts, controls and applets. They may be used simply to enhance the user interface, or contribute to the business logic. The example of the simple business logic is a form validation. In this case, script checks every field in the form to prevent incompatible input before the page is sent to the server. Scripts can also respond to various events (user interactions and browser events), which gives them ability to define the behavior of the user interface (page).

Sometimes, page contains Java Applets or ActiveX controls. These controls and applets can work independently of any scripts in the page, but can also communicate with them (form of interdependency).

Scripts have access to the Web page content via Document Object Model (DOM) interface [2]. This interface is a W3C standard. At the core of the Document Object Model is a set of interfaces that specifically handle XML documents. XML is a flexible language that enables designers to create their own special purpose tags. The DOM interface enables client scripts to access XML documents through ActiveX controls or Java Applets. The main disadvantage of using controls or applets is requirement for the additional software to be installed. In the past, when every browser had its own logic of accessing page contents, thick client pattern was not popular. The reason was that developers had to write different code for every browser, which resulted in Web sites working only on the most popular browser at the moment.

Today, as DOM interface standardized page contents access, future for the thick client got brighter. Moreover, processor power of the client machines has increased, enabling more complicated scripts to be executed on Web client.

We must state that term "rich client" as a middle layer between thick and thin client patterns is becoming popular lately. Its purpose is to distinquish above mentioned type of thick client from the clients working independetly from Web browsers – standalone applications which must be installed on each client system.

V. Proposed Web Client

As World Wide Web Consortium defined several important standards (DOM, CSS, HTML 4.0), we can say that browsers war calmed down. These standards united browsers in some way, defining how to write code which will run on every browser (DOM compatible). They are also the baseline for proposed Web client.

Current client-server communication can be presented on the diagram on Fig. 2:

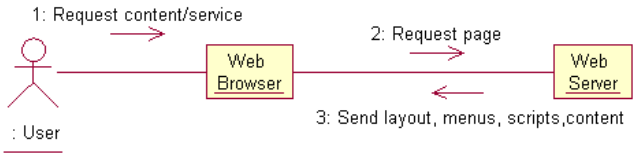

Fig. 2 Common client-server communication model

User is requesting content or a service through some form of interaction (clicking on the link). After that, Web browser will send a page request to the Web server. Server will gather complete page code (layout, menus, scripts and requested content) and send back as a complete page. Problem with this approach is that every time new content is requested, user gets a whole page. Very often, Web content can be only small part of the page. That leads to conclusion that communication is not always optimized. In case of content management systems, with a large number of content requests, this approach makes unnecessary extra network traffic and extra page download time. So, how can we achieve better results?

If Web server could send back only the requested piece of information, that would improve efficiency and decrease network traffic [3] (see Fig. 3).

We propose an approach based on object oriented paradigm. Object oriented thinking is most natural to human thinking. We have developed the framework which acts like a layer between page (HTML) and the Web server. It uses JavaScript syntax, but has its own set of objects.

Purpose of the framework is that developers can think in the more abstract way. Instead of thinking on the level of HTML tags, they can think on the level of objects. To be more specific, this framework recognizes following types of objects:

- resources
- scripts
- contents
- modules

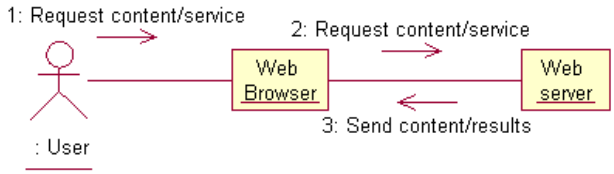

Fig. 3 Improved communication model

Resources are mostly static elements which are used by the Web pages. They can be:

- multimedia files (images, video, audio, Flash, ...),
- menu objects, and
- layout objects

There are two types of menu objects (see Fig. 4):

- menu data objects,
- menu presentation objects, and
- menu controller objects.

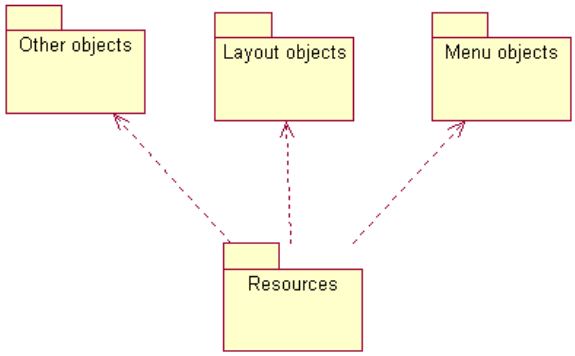

Fig. 4 Resources hierarchy

Menu data object is a structure (most likely tree), which contains all or most of the links to the other pages or options. It does not define the presentation of the menu object (popup, dropdown ...), only its structure.

Menu presentation object defines a visual appearance and behavior of the specified menu data object. In other words, it defines what form will menu data object take – for example popup menu. If a page contains some type of menu (e.g. popup menu), it can be changed to other type of menu, let's say to dropdown menu, even in run-time by calling only one function.

Menu controller object connects one menu data object with one menu presentation object. Together they make MVC (Model-View-Control) framework. Changes in one of them will not influence other two. Also, if we want two representations of the same menu structure, we only have to create two menu controller objects which will be attached to the same menu data object on the one side, and to the different menu presentation objects on the other side.

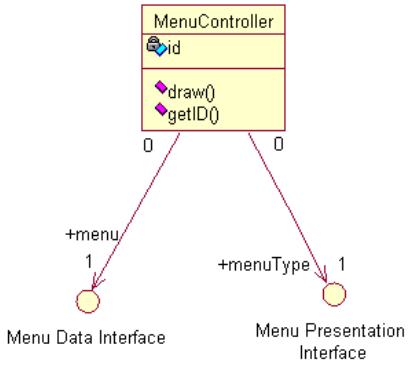

Fig. 5 Model-View-Control framework

Layout object represents visual appearance of the page. It structures the page layout by splitting it onto parts, and it defines their behavior. Those parts (called layout components) can be nested into other parts of the same kind. It is important to say that implementation of the component is abstracted using Bridge design pattern [4]. Than means developers will not need to use tags (<DIV> or <TABLE>) to design layout, but only to define structure by calling layout class methods. Design and behavior of the layout object can be changed in the run-time, simply by calling an appropriate class method. Layout object acts like a "window" to the client – users will experience Web site through it. Thet means we can adapt whole CMS to every Web client only by creating one layout object for each type of client (e.g. for PDA users, users with high resolution displays, etc.).

Scripts define behavior and the dynamics of the Web page. They can be client side or server side. Client side usually defines behavior of the page (user interface and additional layout object behavior, menu object implementations and other). The proposed framework recommends that all client side script code should be embedded into classes, and not as unstructured code. We can clearly see frameworks orientation on object models, and not on data-flow models [5]. Server side scripts dynamically create contents and resources like layout and menu objects.

Contents are actual data the user wants – they can be structured documents (XML, HTML), but also a simple text.

Module is an independent or interdependent component which can be consisted of other modules, resources, scripts and contents. There is an analogy to the folder on the computer file system. It can be independent in a way that it uses only elements from itself, or can be interdependent – if uses elements from the other modules.

VI. Conclusion

By implementing the proposed Web client pattern, we achieved better performance (network traffic and download time) obtained by decrease of the data sent back to a client. Redundant data (data common to all or most of the pages on the site) are not sent, but only desired content. Web client is robust and flexible, since its every element can be changed in the runtime. Here is the list of some possibilities that can be achieved in the runtime:

- Content can be seamlessly loaded or changed in the layout component
- Content can be loaded into every (not only one) layout component.
- Layout design can be changed (in runtime)
- Menu presentation (type) can be changed (in runtime)
- Menu structure can be changed (adding, editing and removing menu items).

Our future work will be based on further integration of the object oriented paradigm into both CMS and Web client, in order to achieve more flexible platform. That would provide advantages like online graphic and layout design and "skinable" Web sites (sites with more layouts which can be switched by one click). These advantages would also enable CMS personalization, modern and very important concept in the process of CMS development.

References

[1] C. D. Cranor, R. Ethington, A. Sehgaly, D. Shur, C. Sreenanz and J.E. van der Merwe, *Design and Implementation of a Distributed Content Management System*, Proceedings of the ACM International

[2] World Wide Beb Consortium, *Document Object Model Level 3*, http:/ /www.w3.org/TR/2004/REC-DOM-Level-3-Val-20040127/

[3] I. Petković: *Component Development of the Client Side of the Web Applications*; Proceedings of 6th International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Services, Telsiks 2003, October 2003

[4] E. Gamma, R. Helm, R. Johnson, J. Vlissides: *Design Patterns*, Addison-Wesley, 1997

[5] I. Sommervielle, *Software Engineering***,** Sixth Edition, Addison-Wesley, 2001