Integration of Object Oriented Web and Centura CMS

Ivan Petković[1], Milena Stanković[2] , Milan Rajković[3], Petar Rajković[4]

Abstract – **Current Web content management systems have limited possibilities while designing frontend layout and user-interface in general. Because of their nature of being general purpose, not every thing that graphic designer has in his mind can be implemented via CMS. We offer a solution to the problem by integrating specially developed client side framework with the rest of the application.**

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

I. Introduction

A content management system is, as the name says, a system used to manage the content of a web site. It typically consists of two elements: the content management application (also known as backend or BackOffice) and the content delivery application (or FrontEnd). 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. Content management systems that are implemented using Web technologies, and where users interact with the system via Web, are known as Web content management systems.

We are witnesses of the enormous number of Web sites. Moreover, that number is growing exponentially. In order to attract users, Web sites must contain some unique features, they must be user-friendly and must have visual identity. In order to achieve visual identity, Web site's client side must be implemented differently than the concurrent's client side. Consequently, we can see various page layouts, Web site graphic designs and user interfaces. Implementing these features requires developing specially tailored code for each site.

On the other hand, content management systems, because of their nature as being general purpose, have mostly predefined format of displaying their content delivery application. That means several different Web sites powered by the same CMS look similar – which is just the opposite from the above-mentioned requirement – to have visual identity.

All authors are with the Faculty of Electrical Engineering, Aleksandra Medvedeva 14, 18000 Nis. Serbia nad Montenegro,
E-mail: {ivanp, mstankovic, mrajkovic, rajkovicp}@elfak.ni.ac.yu

In order to overcome this problem, we developed a framework for Web client development and a Web Content Management System (CMS) called *Centura*. Framework will be discussed in detail later, but it is important to mention that its main purpose is to enable easy development of advanced, visually unique Web clients (Web application's client side). It gives freedom in creating multimedia rich, user friendly and easy customizable Web sites. Since it has simple, butpowerful interface towards the server-side part of the application, it can be easily integrated. On the other hand, Centura is a general purpose, XML-powered CMS that can be used for managing web sites for small and medium sized organizations. We managed to integrate the above-mentioned framework and Centura, and the result was a content management system with the superior user interface and non-predefined FrontEnd display. Centura is still in development phase, but the first test version has already been used for managing the site of the laboratory [1].

The purpose of this paper is to describe the way Centura and the mentioned framework interact [2][3]. In section II we will discuss about the main parts of the Centura Content Management System, and afterwards, in section III, we will describe framework's architecture. Section IV will explain the points of integration between server part of the application and the framework, which resides on the client side. Section V will explain in more detail how the BackOffice section is implemented using framework's capabilities.

II. The Main Parts of Centura

Centura is Web content management system developed using PHP as a server side technology, and a wide spectrum of client side technologies (HTML, CSS, DOM, JavaScript). It is a general purpose CMS which consists of a *front-end*, that is basically a Web site that can be accessed by all users, and a *back office* (figure 1), which is intended for the people who have permission to create and manage content. Although *Centura* is a Web application, its reach client interface reminds of desktop applications. As previously said, Centura consists of FrontEnd and BackOffice. Backoffice provides functionality for managing site content, users, workflow, settings and accounts.

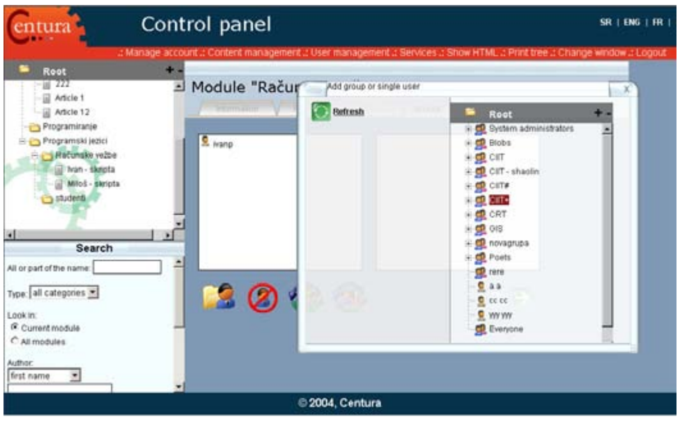

Fig. 1 The BackOffice of Centura

The site content is hierarchically organized using the concept of *modules*, which are similar to folders in operating systems. Modules act as containers for other modules, binary files or items, but more importantly, they are used for setting the access permissions for users and groups. Basic access privileges are *read*, *write*, *delete*, *publish and admin*, but some special-purpose modules can have other predefined privileges, which are a combination of the basic ones.

An item is the key concept of *Centura*, because it represents a web page that will be displayed in the front-end. Items are XML structures that enable creating different types of web pages like *articles*, *courses*, and *polls*. They are created and edited using an online XML editor that is also a part of *Centura's* back-office.

The CMS supports advanced user management and access control. Users are organized into groups that all have administrators assigned to them. Users and groups can be later attached to modules by setting the corresponding access privileges (**Fig. 1**). The concept of the group administrator makes the user management completely distributed, where every group is controlled independently.

III. Framework for Web Client

Purpose of the framework is rapid, quality and cost effective development of the Web application's client side. Its main features are:

- Object oriented approach
- Separation of content, control and presentation
- Elimination of redundancy
- Decreased development time and costs
- More flexible for team work
- Robustness
- Customizable
- Automated and visual development (using CASE tools)
- GUI components management and configuration
- Multilanguage support

Framework provides developers a more natural way of thinking. Instead of thinking on the level of HTML elements, they can think on the level of real world objects, like menus and page layouts. To be more specific, the framework recognizes several types of objects, including controllers, menu objects, layout objects, components, managers and other.

Framework is consisted of several subsystems, each assigned for distinct set of tasks. Following is the list of the subsystems:

- core, main subsystem that acts like the interface between the other subsystems and the rest of the application (e.g. server-side section)
- layout management subsystem, for manipulating, adding and removing page layouts. Every page layout (design) consists of one or more layout components
- navigation subsystem, whose primary purpose is to manage menu structures and the way how they are displayed
- windows management subsystem, which contains logic for creating, removing and managing windows in all other ways
- document management subsystem, with the main purpose to integrate all documents on the Web page into unique entity, therefore enabling interaction between them
- component management subsystem, for supporting and managing user interface components. Each component implements interface that provides customization, graphic and dictionary support (**Fig 3**)

As you can see on **Fig. 2**, Controller is the main component of the framework's core. Hence, all the communication between the external sources and the rest of the system is going through controller. This architecture provides minimal redundancy and optimal communication between framework and other systems.

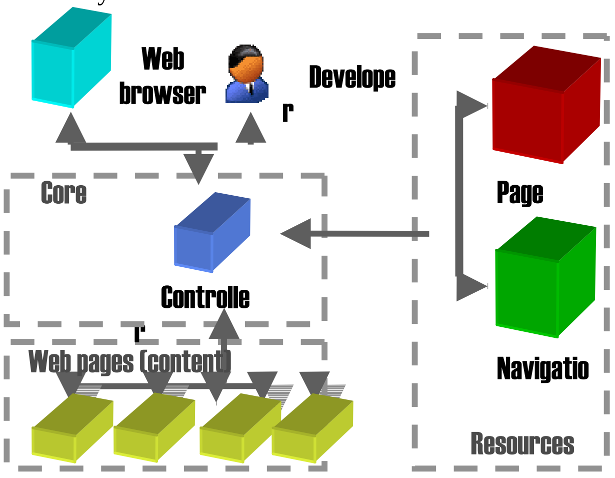

Fig. 2 Architecture of the Web site using the framework

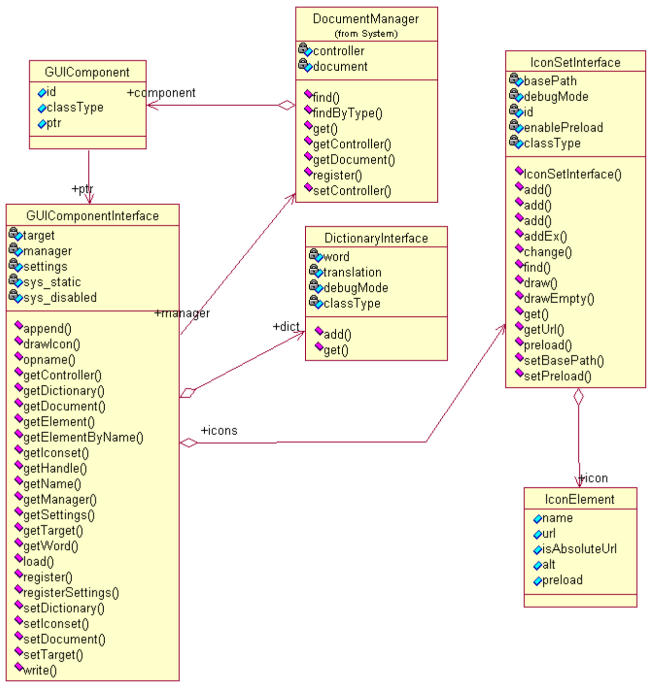

Fig 3 Framework's component class diagram

IV. Point of Integration

Since the framework was used for design and development of the entire client side of the BackOffice, we can sort its application into several categories:

- navigation development
- design and layout
- document manipulation
- management of components and managers

First three applications are described in the previous section. Last one represents extended functionality of the framework, as many components and managers were specially developed for the BackOffice purposes. Furthermore, framework's extendible architecture provides only solid basis for developing objects that are more advanced. As stated before, there are two types of objects used for extending framework's basic functionality – components and managers. Both components and managers are classes derived from *GUIComponent* base class. We can see from the class diagram shown on **Fig 3** that DocumentManager-class object knows about all registered components in the document. Important to say is that one Web page can contain multiple documents (stored in frames), and that framework defines only one DocumentManager-class object for each document on the page. Similar to Fascade design pattern [6], these managers possess logic how to manipulate components, and they act like an interface between controller object and the registered framework components. Document managers are connected with the Controller object, therefore creating the network of document managers with all the capabilities to interact with each other.

On the other hand, separation between dictionary object (DictionaryInterface), icon set object (IconSetInterface) and component (GUIComponentInterface) is crucial for the implementation of the concepts like customization, personalization and multilanguage support. In addition, these architecture represents the interface between client-side and server-side developers. Implementation files (definitions) of these classes are loaded in the beginning, but the setting object and dictionary object are defined by the server-side developers, who can, using this mechanism, change component's appearance or even add a new language.

V. BackOffice Design

Centura's BackOffice is quite different from the BackOffice sections of other Web CMS. Instead of having a standardized Web-based logic and layout, it possesses appearance and functionality of the desktop application. This provides much better "look and feel", a crucial factor when evaluating user-friendly environment.

As we can see on **Fig. 1**, BackOffice has Windows Explorer style layout. That means the screen area is divided into three main parts:

- Main menu and options area (on the top of the page)
- Navigation menu (on the left)
- Content (central part)

Navigation menu is implemented as a dropdown menu. Corresponding menu controller use dynamically created menu structure (*MenuDataInterface* derivate), and a class which implements dropdown menu (*MenuPresentationInterface* derivate).

Content area is loaded after a user selects a menu item (from the navigation menu). Framework supports seamless loading of the external pages into layout components, while the rest of the page stays intact. On contrary to the usual Web logic, rest of the page is not reloaded, which significantly decreases the download time. Loaded page is linked to the controller object, and therefore integrated into the system. Using this feature, several documents can interact with each other in a very simple and efficient way.

Seamless loading and multiple document manipulation happened to be extremely useful for purposes like selecting users, groups or items from the existing hierarchy. For this purpose, class named *ItemManager* has been designed and implemented. Menu structure is dynamically loaded only after the user requests the object of that class. On **Fig 3** we can see the *ItemManager* in action (in the opened window).

VI. Conclusion

Integration between the framework and the server side of the Centura CMS resulted in many significant improvements and benefits. Flexible and extendible architecture of the framework enables unlimited possibilities when designing Web site layout and user-friendly interface.

Our future work will be based on further integration of the framework and Centura CMS, in order to achieve even more flexible platform. That would provide advantages like online design of the page or template layout and run-time customization and personalization of Web sites.

References

[1] http://ciitlab.elfak.ni.ac.yu

[2] I. Petković, M. Stanković: *Object Oriented Web Client for Content Management Systems*, Proceedings of ICEST 2004, Bitola 16-19 June 2004, 289-292

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

[5] 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

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