The CONED project: a WEB-based Virtual University

Tudor Marian and Bogdan Dumitriu

Department of Computer Science
Technical University Of Cluj-Napoca
Baritiu 26, Cluj-Napoca, Romania

tudorm@coned.utcluj.ro and bdumitriu@bdumitriu.ro

Abstract. *A virtual learning environment represents a space available on-line where both students and teachers are brought together to interact similarly to the way they do in reality. Although efforts towards the materialization of this concept are currently on track, the domain has yet a long way to go before reaching a mature state. Consequently, our work tries to make a contribution by means of study and practice of e-learning. This paper tries to bring forward the aspects that individualize such an environment and to present the particular solution developed in order to meet its requirements. We mainly focus our attention on the specific architectural design on top of which this solution was built.*

Keywords: *virtual university, academic environment, Java, distance learning, open architecture, 3-tier architecture.*

1 Introduction

It is becoming increasingly clear that the future is *"e-"* prefixed, making it only a matter of time before most traditional domains will be doubled by their *"e-"* equivalent. Since the academic environment is no exception, it is only natural that technological support is required for aligning it to this tendency. The first thing that comes to mind when talking about universities on the Web is e-learning. An e-learning environment is supposed to provide both teachers and students with a way of carrying out common educational activities within virtual premises.

One of the things we were interested in, as far as e-learning is concerned, was to create an application by means of which we could allow teaching to take place through the Internet. We designated such an instrument as *virtual university*. In short, the virtual university intends to closely replicate a real academic environment, with its main purpose being to offer the possibility of remote education to those interested. Given the nature of the domain we based our application on, it was vital for our design to be a very flexible one in order for it to support regular modifications and additions in the years to come.

After a careful study of the available options, we reached the conclusion that Java technology is the best choice for developing our virtual university. We had several reasons to guide us in our decision: Java allows portability to our application, the Java platform benefits from powerful enterprise support and it is free technology – an important advantage in the academic environment.

In the sections to come we try to indicate how our project evolved by describing the general UML design steps we followed as well as the way we used Java in our implementation. We begin this in section two with the early, conceptual design stages: identifying the requirements for our virtual university and choosing the technologies to use. We carry on in section three with an enumeration of the services we wanted to make available through our environment and the presentation of the design goals. Section four deals with the architectural model of the application, putting the emphasis on the way it helped us stay within our initial constraints. We end the discussion with some conclusions and possible developments in section five.

2 Fundamentals

The task of exploring the web-based education area was initially started at our university as a Tempus project [5] by a research group. This group has closely looked into the various aspects regarding distance learning and eventually produced a material based on its study, namely [1]. At this point, as the theoretical part was coming towards its final stages, a practical design and implementation were required in order to materialize the results of the research. Taking on from here we would like to present the way in which this was approached and completed.

Premises

What we intended to achieve was a web-based application which could allow the previously identified concepts to be put in practice. This meant that we wanted to offer a frame in which the distance learning (defined in [1] as "an instructional strategy that accommodates learners who cannot attend classes due to time or place restrictions") process could take place. Our web-based university should bring teacher and student together in a virtual environment so that learning activities could take place more or less in the same way they do in reality. This would basically imply offering a means for both teacher & student to have access to course material, communication services and a way in which assessment and grading can take place. The final outcome of this process would be to provide students with a course certification at the end of their studies.

Technologies

The analysis of the functional requirements of our application (web clients, large amount of data to work with, the need for an attractive, yet simple, interface) led us to the conclusion that what would best suit us is a 3-tier architecture. Such architecture would allow us to achieve a high level of independence in implementing our design; the use of a wide range of resources would thus become possible. The following figure best illustrates this concept:

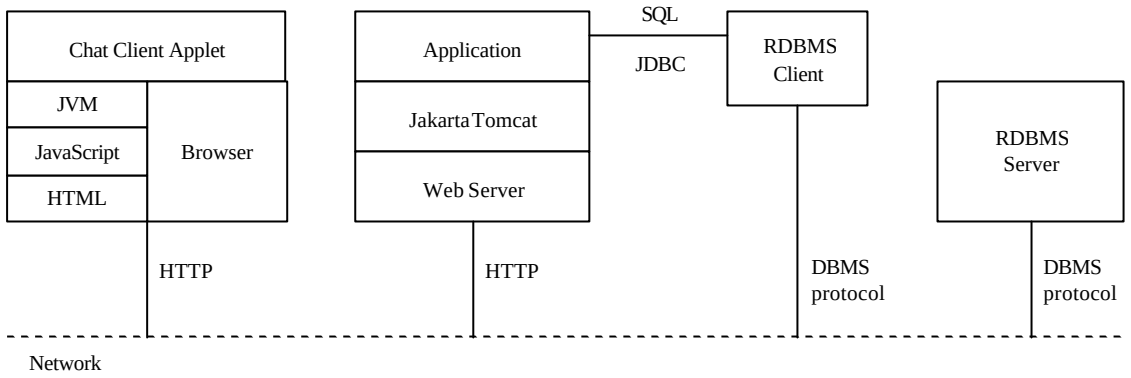

Fig. 1

The entire first tier (encapsulating the presentation level), represented by the client desktop on which a Web browser is running, is based on html. Additionally, it provides support for JavaScript as well as Java applets.

As far as the middle-tier (encapsulating the business logic) is concerned, our choice was to rely on JavaServer Pages [6] and Servlet [7] technology and to use Jakarta Tomcat [8] server as support. We decided on using JSP and Servlet technology in order to bring the power of Java into our application. The choice of using Jakarta Tomcat, on the other hand, was determined both by its close relation to the widely-used multi-platform Apache web server (thus assuring maximum efficiency) and by the fact that the former's code base is included by Sun in the J2EE reference implementation.

Finally, the back-end tier (comprising of the data layer) is ensured by one or more database engines. Our choice for relational database management system (RDBMS) was Microsoft SQL Server 2000. The Java Database Connectivity Data Access API provides us with a means of connecting the middle to the back-end-tier in a very flexible manner, thus making the replacement of the RDBMS very easy.

3 Functional and non-functional requirements

Although the guidelines for our application were clearly defined by the results of the above mentioned research, we considered that analyzing the functionality offered by similar existing products would be of aid in shaping the offered services. Visitor level access to the *University of Maryland University College* (UMUC)'s virtual university [10] was used for this purpose. As a result, the boundaries within which our virtual learning environment should be modeled became slightly more accurate.

Once the initial steps were completed we shifted our focus onto identifying the services that should be made available through the application. The outcome is summarized in the enumeration that follows:

- **static materials presentation** (presentation of course specific materials)

- **dynamic material presentation** (presentation of class specific materials)
- **conference services** (area where topic based message exchange takes place)
- **chat room services** (area where real-time general purpose conversation takes place)
- **email services** (a means of contact among members of a class – teacher and students)
- **announcement services** (a means for the teacher to announce class related events)
- **task assignment** (a means for the teacher to assign tasks to the students)
- **grading book** (area where the teacher can award grades to his students)

Additionally, a certain number of lower level sub services have been defined, among which the upload service, allowing both teacher and students to attach files to various entities such as tasks, messages and so on.

Three distinct goals were established from the very beginning of the design process. The reason behind this was the desire to develop an application adaptable to a wide range of contexts and to constantly evolving demands. The first of them was **independence**. In addition to the technological decisions explained above, all taken with this specific goal in mind, we also tried to define our architecture in such a manner so that it would allow all of its composing subsystems to be as self-consistent as possible. Secondly we aimed **configurability**, the achievement of which was accomplished by making extensive use of eXtensible Markup Language (XML) technology in the business logic layer. Last but not least, we wanted **extensibility**. The architectural core was oriented towards this specific demand and the way it meets it will become clear in the section to follow.

The final decision taken before starting the actual development was to adhere closely to the Unified Modeling Language (UML) process of creating an application, as described in [2] and [3]. This allowed us to organize the work in a very systematic manner, to precisely identify all the requirements, to create powerful analysis and design models and eventually come up with a successful, extensible and configurable implementation. Furthermore, the use of the unified process created the possibility of specifying a scalable, open architecture from the very early stages of development, thus greatly simplifying the following phases.

4 Architectural model

Once the use case and analysis models [2] were completed, the project advanced to the next step: its design. This meant an architecture capable of supplying the support for all the functionality defined in the previous stages had to be created. Before modeling such an architecture ourselves, we examined some of the existing work in this direction and focused our attention on a particular project, namely Struts [9] which offers a very scalable, open source solution for rapid development of web applications based on JavaServer Pages and Servlet technology. Even though using Struts could have been an option, eventually we decided against it due to the fact that time was not a constraint in this case and, under these circumstances, an architecture designed precisely on top of our models would work better than a more general one.

The following figure is an UML class diagram that offers an overview of the structural design. What stands out is the clear separation between the interface, business logic and data tiers.

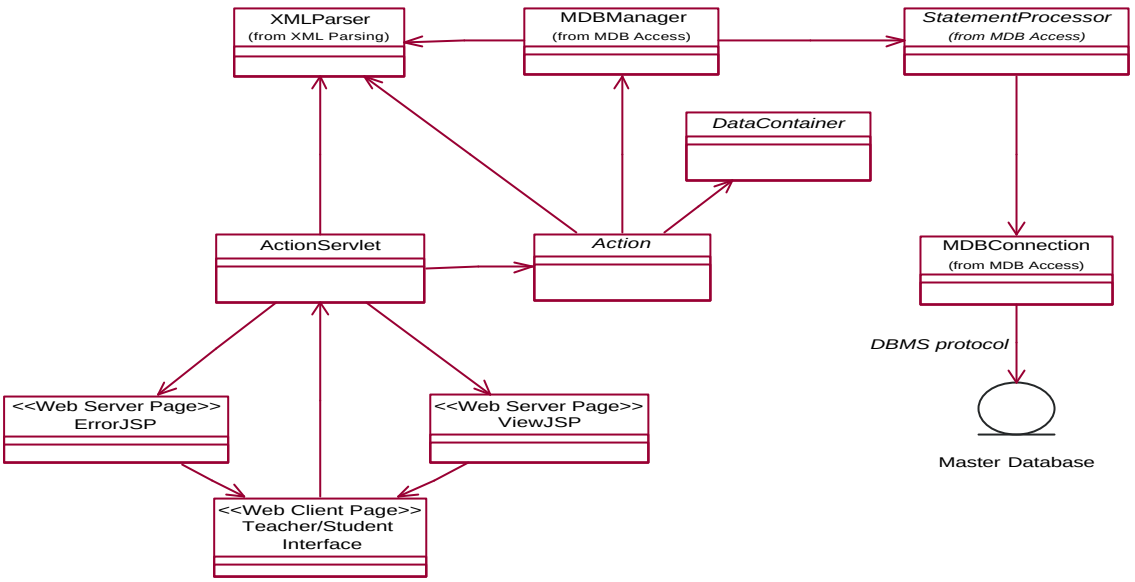

Fig. 2

The process is initiated by web clients sending requests to our system. These are all handled by a central dispatcher servlet (the *ActionServlet*) which will choose the appropriate *Action* class for further processing. The choice is based on URI-to-action mappings stored in an XML configuration file. It is the *Action* class' task from here on to properly use the database access layer in order to either deposit or retrieve user data and modify if the case arises. Once this is finished, a response is returned to the *ActionServlet* which in turn forwards it to a JSP page run by the server, thus generating an adequate response for the client. Should an error occur in the process, however, a configurable JSP error page is called. Action specific JSP response pages are configured via the same XML file as above.

Extensibility is ensured through this design by means of the XML configuration file we spoke of earlier. All that is required for adding new behavior to the system is for the developer to create the handler classes and introduce their names in the configuration file. An example of how the architecture functions for a particular case (the one where a teacher wants to add a new conference in the forum section of the university) can be analyzed in the sequence diagram to follow:

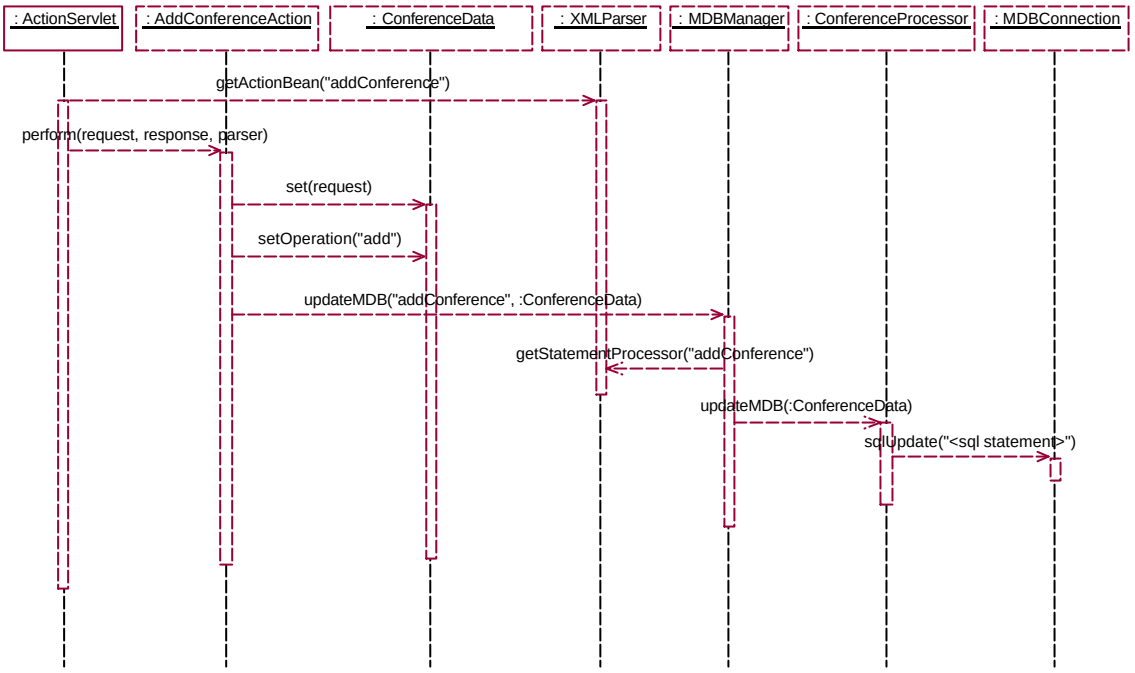

Fig. 3

A distinct module, not illustrated in the architectural model above, has been designed to meet security demands. This module is defined in terms of restricting access to various areas of the university for certain types of users, completely configurable via a stand-alone XML file. Upon initialization, this file is processed and the established security policy is enforced.

The entire implementation was thusly built on top of this architecture, in more or less the same way Fig. 3 illustrates, allowing all the services described in section 3 to be made available through a unified interface.

5 Conclusions

This paper presented a system created to comply with the ever increasing demands of web based education, the services it provides and the architectural model created during its design phase. The virtual university application developed did indeed meet the goals set at the beginning, namely to provide a materialization of the theoretical results obtained in the research process. In addition to this, it also proved itself to be a viable system, ready for deployment.

The architectural model, specifically conceived to anticipate constantly changing requirements, backed by Java technology, formed the basis of straightforward development and currently ensures the premises for ongoing efforts of extension.

We consider further development in two directions: completing the already existing services of the virtual university with additional ones and creating a separate module which would allow remote

access to the administrative tasks regarding system management. Work on the latter is currently in progress.

References

[1] Anonymous authors, "Web-based Educational Technology", Casa Cartii de Stiinta Publishing House, 2000.
[2] Booch G., Jacobson I., Rumbaugh J., "The Unified Modeling Language User Guide", Addison-Wesley Pub Co, 1998.
[3] Fowler, Martin, Scott, Kendall, "UML Distilled: A Brief Guide to the Standard Object Modeling Language (2nd Edition)", Addison-Wesley Pub Co, 1999.
[4] Forta B. et al, "JavaServer Pages Application", Teora Publishing House, 2001.
[5] Tempus home page, http://www.etf.eu.int/tempus.nsf
[6] JavaServer Pages home page, http://java.sun.com/products/jsp/
[7] Java Servlet technology home page, http://java.sun.com/products/servlet/
[8] The Jakarta Tomcat project home page, http://jakarta.apache.org/tomcat/index.html
[9] The Struts home page, http://jakarta.apache.org/struts/index.html
[10] University of Maryland University College start page, http://www.tychousa.umuc.edu