Novi Sad J. Math.
Vol. 37, No. 2, 2007, 181-192

FACULTY INFORMATION SYSTEM BASED ON OPEN SOURCE TECHNOLOGIES

Srđan Škrbić[1], Žarko Bodroški[1], Biljana Pupovac[1], Miloš Racković[1]

Abstract. In this paper we present the use of open source tools and technologies in information system development process at the Faculty of Sciences in Novi Sad. We describe specific conditions in which this information system is being developed, and present its architecture in detail. In addition, we give an overview and present the way in which we used the most important technology in the implementation of this system - EJB 3.0 and describe specific open source tools used in the implementation processes. Finally, we present our conclusions and generalize them to all information systems developed in similar environments.

AMS Mathematics Subject Classification (2000): 68N01

Key words and phrases: information system architecture, EJB3, web services, Java

1. Introduction

The Faculty of Sciences, one of the largest faculties of the University of Novi Sad employs over 400 staff and has substantial IT requirements. Moreover, exchange of data in electronic form with other faculties, the Rectorate and Ministry offices has become a must. The IT environment at the Faculty has heterogeneous application platforms and associated servers, mainly implemented in obsolete technologies. To address the IT needs, the Faculty is developing and deploying an open source information system.

There are many reasons why open source platform has been chosen. Free and open source software has been used to create many of today's most innovative products and solutions [3]. Nevertheless, an element of risk in deploying open source based solutions has been perceived. Open source systems lack the comfort zone that a commercially acquired solution provides; rather, support comes from bulletin boards and the like.

However, open source software has been widely used for a long time in academic institutions. The Faculty of Sciences in Novi Sad is no exception. As a consequence, IT staff and researchers have valuable experience in open source software usage and maintenance. In this way, the usage of this type of software comes with drastically lower cost than usual. Also, because open source software is more or less free, there is often the misperception that service and support

[1]University of Novi Sad, Faculty of Science, Department of Mathematics and Informatics, Trg D. Obradovića 4, 21000 Novi Sad, Serbia,
e-mail: {shkrba, bodroskiz, biljanap, rackovic}@uns.ns.ac.yu

should also be correspondingly priced, which is a difficult mindset to break. Although the open source phenomenon is sometimes characterized as a threat for the software development industry, small to medium-sized enterprises anywhere in the world leverage the innovative open source model as an infrastructure on which to create new business opportunities [6]. Being the most complete and most popular open source development technology, Java platform is used.

In this paper we give details about the architecture of the information system and describe specific technologies and products used in software development and production. In the following section we describe conditions in which this information system is being developed. In addition, we explain architecture of the system devised to allow successful development in this environment. The third section describes technologies used to implement the system according to constraints and goals set by its architecture. In addition, it contains a brief retrospect about lessons learned in this new field. The fourth section introduces specific tools used to develop and deploy components of the information system together with the discussion and reasons why these products are used. The final section is the conclusion.

2. Architecture of the Information System

The Faculty of Sciences, being an institution that, among others, educates computer scientists and practitioners, has enough human resources to develop the information system on its own. However, the structure of IT experts is heterogeneous. They are divided in independent teams that streamline research and practice according to their interests. That is why it was decided that every team will develop one large segment of the system. These segments do most of their work on their own, but communication between them is necessary and vital for implementation of integrative requirements.

In order to divide effort to individual teams, the information system is segmented in ten segments:

- student service,
- accounting,
- human resources,
- library information system,
- teaching and e-learning,
- fixed assets,
- research project management,
- document management,
- business intelligence.

First four of these segments are currently under development. Three types of clients that the system should serve are foreseen. The first type consists of Swing applications developed to be used by staff at the Faculty. Clients that access the system using web browser fall into the second type. Third type is related to external SOAP clients - other information systems, especially the one in the Rectorate, that connect to the system using web services.

Implementation of one more segment of the system is foreseen as one of the last tasks. It is a web portal, a component that integrates all web related services offered by other segments in one easy searchable web site.

Described environment has had critical impact on the information system's architecture. Being a modern information system, it is based on the multi-tier architecture that allows all three types of clients to be served. Communication between individual segments is being implemented using secure web services.

A Graphical representation of the information system's architecture is given in Figure 1. For the sake of simplicity, this figure shows only three segments of the information system – student service, accounting and human resources. Other segments are connected to the whole using the same principle.

Every segment is a three-tier application for itself. It consists of database management system, application server, and a set of client GUI applications. This construction is well known, although the one with web clients is more popular. In this case, GUI clients are needed because of the complexity of the processes conducted by the Faculty services and overall performance.

The application server in every segment of the system is divided in two layers. The first one is business logic of a particular information system's segment, while the other is a web container. GUI clients communicate directly with the corresponding business logic layer, while external clients – web browsers and SOAP clients, communicate with the web container.

It is also worth to noting that individual segments also communicate with each other using web services. This communication differs from the communication with SOAP clients in two ways. First of all, this is communication between business logic layers, and not between a SOAP client and web container. In addition, business logic layers of other segments of the system are allowed to access a much larger set of services than external clients.

Moreover, modern web 2.0 applications running in web containers are set as a requirement. Web clients communicate with web container using JavaScript, asynchronous calls and XML as a medium for data transport.

Two problems arise in an attempt to implement described architecture using open source technologies. Are there open source tools capable to implement this architecture? If there are such tools, it is a question how do they perform compared to their commercial counterparts?

After researching many possibilities, we come to a conclusion that there are open source tools that allow efficient implementation. The second question is more difficult to answer. In an attempt to do so, we will present specific tools, servers and operating systems used in our implementation. In the following sections we describe implementation of this architecture and technologies used in that process.

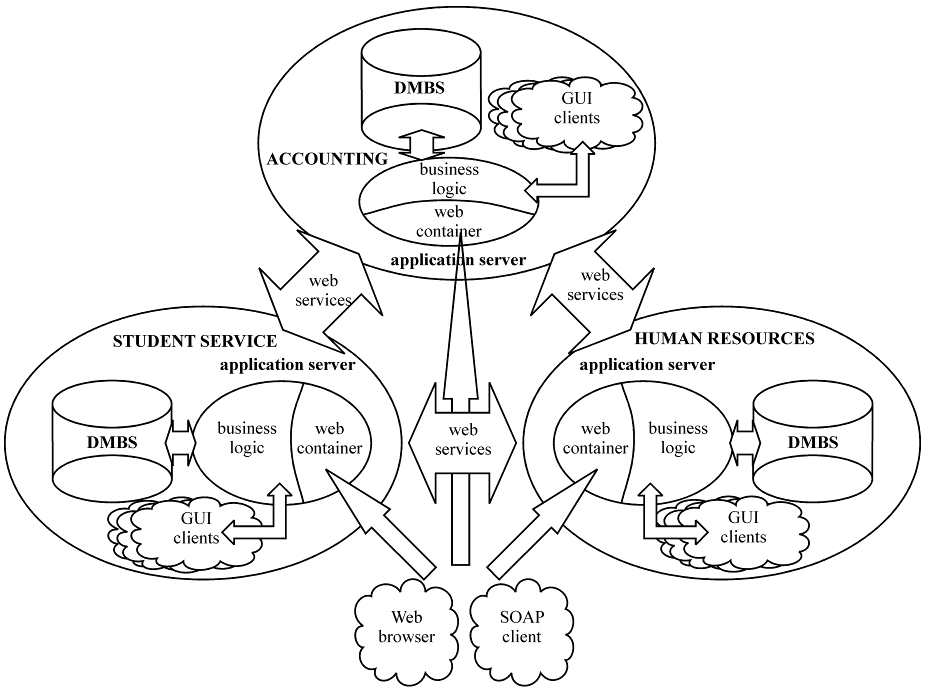

Figure 1: Architecture of the Information System.

3. Implementation

The main technology we use is the Enterprise JavaBeans 3.0 (EJB 3.0) [5] implemented by the JBoss 4 application server. This technology covers most important tasks required by the system architecture. EJB components map database schema and allow business logic to be implemented. Moreover, support for web services is a part of the EJB 3.0 specification. JBoss application server [10] supports EJB components and offers a web container capable of serving JSP (Java Server Pages) and JSF (Java Server Faces) pages. The EJB technology and JBoss application server and their capabilities are described in more detail in the following two sections.

A good way to introduce a technology is to give a definition of its inventor. Sun Microsystems' definition of the Enterprise JavaBeans architecture is as follows:

The Enterprise JavaBeans architecture is a component architecture for the development and deployment of component-based distributed business applications. Applications written using the Enterprise JavaBeans architecture are scalable, transactional, and multi-user secure. These applications may be written once, and then deployed on any server platform that supports the Enterprise JavaBeans specification [5].

This long and a bit confusing definition means that EJB is a server-side component that can be deployed in a distributed multi-tiered environment. If we look at the most common enterprise applications with traditional four-tier layered architecture, we can place EJB in the middle of this architecture as a business and persistence tier. In that case, the EJB client may be a servlet, a JSP, a standalone Java application, an applet, SOAP-based web service client, or even another EJB.

The main idea of the EJB concept is to easily make distributed application without the need to write complex distributed component framework. In this rapid development the EJB must take care about writing scalable, reliable, and secure applications. From a developer's point of view an EJB is a piece of Java code that executes in a specialized runtime environment called the EJB container that provides a set of component services. The persistence services are provided by a specialized framework called the Persistence Provider [13]. The EJB 3.0 concept will be introduced in two perspectives: as a component and as a framework.

EJBs are server-side components which are used to build parts of the application like the business logic or persistent code. All other functionality behind that code is invisibly managed by the Java persistence API (JPI).

There are three types of EJB components: session beans, message driven beans, and entity beans. Session beans and message driven beans are used to implement business logic in an EJB application, while entity beans are used for persistence. To understand relations between beans, EJB container and Persistence Provider it is necessary to explain EJB components in more detail.

Session beans model business processes, and their primary purpose is to perform actions. Session beans are further divided in two categories – stateful session bean and stateless session bean. A stateful session bean automatically saves bean state between client invocations without having to write any additional code. In contrast, stateless session beans do not maintain any state and model application services that can be completed in a single client invocation [12].

Message driven beans (MDB), like session beans, perform action but the difference is that MDB can be called only by sending a message to those beans. It means that there is no direct access to a method in a message-driven bean.

Entity beans model business data. To make a comparison between session beans and entity beans, the simplest way is to present session bean like a verb and entity bean like a noun. In object-oriented world, entity is an object that can cache database information that represents a persistent content. The definition of entity is older than Java and it still holds true: "An entity is essentially a noun, or a grouping of states associated together as a single unit. It may participate in relationships to any number of other entities in a number of standard ways." [2].

The major difference between entity beans 2.x and 3.0 is in the word "bean". It means that entity bean 3.0 is not managed by container like older version. Instead of that, entity beans are managed by Persistence Provider through the Entity Manager interface [5].

To interact with entity beans, Java Persistence provides a new service called the Entity Manager. All access to an entity bean goes through this service. It provides a query API and life cycle methods for entity beans. Unlike older versions of the EJB specification, in Java Persistence, entity beans and the Entity Manager do not require an application server to be used. Java Persistence can be used in unit tests and standalone Java applications like any other Java library. This arrangement of components is represented in Figure 2.

Session bean and message driven bean run in a different environment than entity beans, and they are completely managed by EJB container. It acts as an intermediary between the bean and the EJB server. The EJB container interacts with EJB objects and provides services such as security, transactions, concurrency, and naming at runtime [1].

In the second section of this paper a preview of technology was made and a brief look at component organization is given. In this section we explain some of the benefits of the new concepts in EJB 3.0 relevant to the information system.

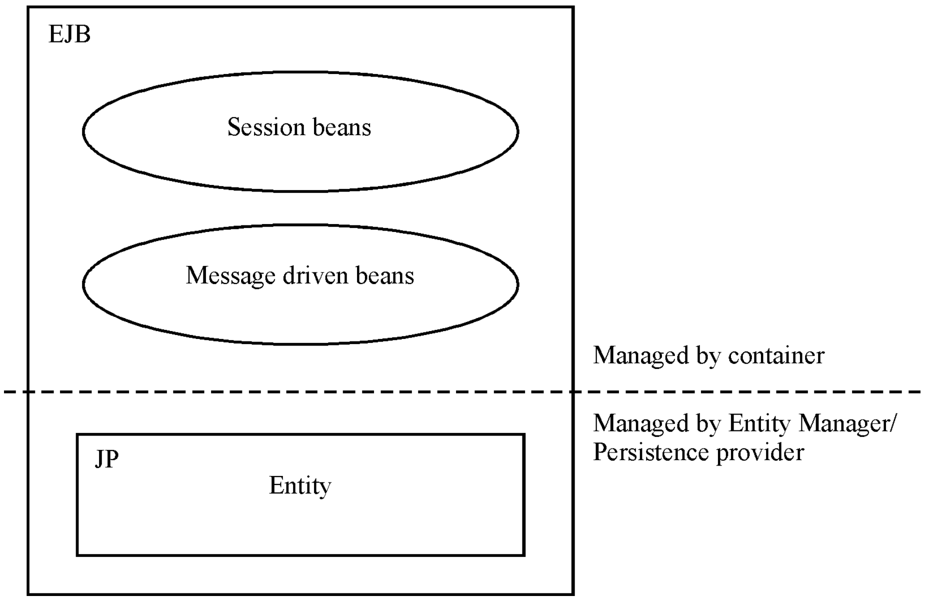

Figure 2: Overall organization of the EJB 3.0 API.

In order to explain the benefits we take a closer look at some major problems with the older version of EJB:

- the need to directly use Java Naming Directory Interface (JNDI),
- verbose XML deployment descriptors,
- a heavyweight programming model.

Techniques introduced in EJB 3.0 eliminate these sources of complexity. They are as follows: metadata annotations, minimal deployment descriptors and dependency injections.

Annotations are custom Java modifiers that can be used by anything handling Java source or byte code, and it allows entering additional attributes (information) to a Java class, interface, method or a variable. With this functionality, Java code is still POJO which makes programming model much simpler, and eliminates the need for writing detailed deployment descriptors. Still, it is possible to mix annotation with deployment descriptor by overriding an annotation.

These techniques solve the problems with deployment descriptors and drastically reduce weight of the programming model. One more technique has the same effect on complexity of code but, more importantly, it introduces a way to cope with the JNDI problem. This technique is called dependency injection. It is some kind of a reverse JNDI lookup. In the previous model of JNDI usage, the bean explicitly retrieves resources and components it needs. As a result, component and resource names are hard-coded in the bean. On the other hand, with dependency injection the container reads target bean configuration, figures out what beans and resources the target bean needs and injects them into the bean at runtime [12].

Apart from services and techniques which solve major problems in older versions of EJB, an important role in the development with EJB 3.0 has teh Java Persistence Query Language (JPQL). JPQL is based on EJB Query Language (EJBQL) and was first introduced in EJB 2.0. It is a portable query language designed to combine the syntax and simple query semantics of SQL with the expressiveness of an object-oriented expression language. This is a language for querying entity beans and representing queries in terms of entity beans and their relationships [9].

After the implementation of EJBs as a middle tier of architecture a logical step was to use JAX-WS 2.0 (The Java API for XML-Based Web Services) for web service client development. This specification is a Java EE 5 extension of the Java API for XML-RPC, and the goal of this technology is to simplify the development of web service applications. Both a Java web service and an EJB web service support dependency injections and lifecycle methods, but there are a few more reasons why we decided to use the EJB web service:

- benefits of declarative transaction and security which is automatically available from EJB 3 components,

- web services that use EJB 3 can easily expose business application using additional protocol by adding a remote interface,

- simplified development process by dynamical processing the EJB annotation during deployment.

Let us consider the StudentDataWebService offered by the student service segment. For a given student book number it returns various data about student. For example, library information subsystem, as one of the segments of the integral information system can use this web service to obtain basic data about a particular student. At the same time, the middle tier of the accounting subsystem can use the same web service to obtain call the to number for a given student. In addition, external SOAP clients can use this web service to obtain necessary information. In this way, no information is stored twice or more times in two or more segments. If a segment needs to obtain data from the other parts of the system, it uses web services.

If data is exchanged in this manner, security of the system has to be carefully addressed. Even from this aspect, the EJB 3 technology offers a solution. Security and access privileges are added to every EJB method, including web services. The EJB container reads these privileges and controls access. It is integrated with the LDAP server with data about all network users maintained at the Faculty of Sciences, so the existing users and groups can use the already assigned credentials.

4. Selection of tools

For the purpose of data storage, the MaxDB version 7.6 DBMS has been chosen. The MaxDB is an ANSI SQL-92 compliant relational database management system.

Upon selecting of DBMS, the following characteristics (also emphasized by MaxDB manufacturers themselves), have been considered [11]: easy configuration and low administration, elaborate backup and restore capabilities, continuous operation, no scheduled downtimes required, designed for large number of users and high workloads, scales to database sizes in terabytes, etc.

Beside mentioned reasons, this DBMS is chosen because it features the "Oracle mode" that allows communication with the database using Oracle variation of the SQL language. Previous prototype version of the student service information subsystem used Oracle DBMS, which had to be replaced when transition to open source platform begun. On the other side, the EJB components running inside the application server are the only components that communicate with databases, and they are independent of specific DBMS. This directly means that this information system can use any DBMS.

The Eclipse is a universal tool platform [4]. Its main characteristic is that it is based on plug-ins that implement different functionalities. Existing tools in the Eclipse can be extended by adding new plug-ins. The Eclipse includes the basic platform plus two major tools: Java Development Tools (JDT) and Plug-in Developer Environment (PDE).

The functionalities implemented as plug-ins provide possibilities for a complete development of the information system using Eclipse. One of the plug-ins that is used during the development is a WTP plug-in. This plug-in is a result of the Eclipse Web Tools Platform Project and it provides APIs for the development of J2EE and Web applications. The subproject of the Web Tools Platform Project is the Dali JPA Tools Project. As a result of this project, a Dali plug-in is implemented. This plug-in provides support for object-relational mappings for the EJB 3.0 entity beans via the creation and automated initial mapping wizards and programming assistance such as dynamic problem identification.

After careful consideration of other open source alternatives, JBoss AS 4 is chosen as an application server. It is an open source J2EE-based application server implemented in Java. Accordingly, it is usable on any operating system that Java supports.

At the center of the JBoss is a JMX (Java Management Extensions) microkernel that manages the managed beans (MBeans) that control the various services in the server [7]. Each of the J2EE services that run inside JBoss is an MBean. Services such as Log4j, Tomcat, and Hibernate are all MBeans.

Beside the mentioned, JBoss supports a variety of J2EE services:

- Enterprise JavaBeans 3.0,
- Java Persistence,
- Java Message Service (JMS),
- Java Transaction Service/Java Transaction API (JTS/JTA),
- Servlets and Java Server Pages (JSP),
- Java Naming and Directory Interface (JNDI).

Owing to the fact that JBoss provides support for EJB 3.0 and that it is the leading application server in the open source application servers domain, we chose it for the deployment of information system's components.

JBoss also provides support for Web application deployment via Tomcat JSP engine included as a service in JBoss. This service supports Servlet 2.4 and JSP 2.0 specification. Since Tomcat is integrated in JBoss, it is possible to develop and deploy applications that combine J2EE and Web technologies. The implementation of the student service information subsystem is actually performed by combining the J2EE and Web technologies, which has been an additional reason for choosing JBoss as an application server.

Additionally, JBoss has a clustering support. Clustering provides possibilities to run the application on several parallel servers, the so-called cluster nodes. In JBoss, the nodes are JBoss instances and a cluster is a set of nodes. Clustering is important for a stable work of enterprise applications. Moreover, performance improving can always be done by adding more cluster nodes, which adds to the scalability of the system.

Choosing Eclipse which has been used in the implementation of the entire information system, and JBoss application server that supports J2EE application has also proven to be suitable because of the simplified application deployment process. Eclipse provides support for automatic generation of ear files that contain all specified modules listed in a suitable configuration file.

Services that run inside JBoss, which provide deployment of th applications developed by combining several various technologies, EJB 3.0 support, clustering support and easy deployment, as well as JBoss proven stability, have completely satisfied the needs of the information system.

For implementing GUI clients we use the most logical solution regarding previous choices – Java and Swing platform. Using any other Java technology for GUI development, like SWT (Standard Widget Toolkit), would not have made any difference. As we mentioned earlier, the web applications offered by the system are required to use asynchronous calls, JavaScript and XML. This type of web applications is called AJAX (Asynchronous JavaScript And XML). In order to implement these functions in the most efficient way, we use GWT (Google Web Toolkit) [8]. GWT is an open source, Java-based framework for creating Java-based AJAX web applications. This tool makes writing web applications similar to Swing applications. Moreover, Eclipse plug-in Cypal Studio for GWT simplifies development of these applications even further. In this way, the developer is only concerned with Java code, and does not have to develop or maintain a mixture of HTML, JavaScript and Java code.

5. Conclusion

We present a robust and flexible architecture of a modern faculty information system tailored to satisfy primarily IT needs of the Faculty of Sciences in Novi Sad, although it fits well to any other faculty at the University of Novi Sad. This model includes the usage of most up-to-date patterns and mechanisms, such as multi-tiered model, web services and AJAX. The latter suggests that it supports and maintains communication with inner as well as remote clients.

The architecture of the information system allows independent development of specific segments in heterogeneous technologies. The segments are then integrated in an enterprise system using web services. Although it was planned to use only open source technologies, it is possible to implement some segments using other platforms that support web services integration.

A successful attempt was made to implement such an architecture using only open source software. As a result, specific technologies and software tools used are listed and described. The idea was to give an insight into what open source tools and technologies exist today, and how useful are they in the process of building an enterprise IT project. On the other side, the same architecture is well usable and implementable using other commercial technologies and tools.

Experience gained in the information system planning and development process leads to a conclusion that these tools are more than sufficient to build large systems. Newest technologies and specifications, such as EJB 3.0, contribute to this statement even further, and drastically simplify the development and deployment process. It is also worth of noting that open source software, such as the JBoss application server and Eclipse, is not only comparable to commercial products, it is superior to them in many aspects. To support that fact, we will mention that the first and, up-to-date, the only implementation of EJB 3.0 specification comes from JBoss, making this application server a technological leader in this area.

Acknowledgement

This paper is part of the scientific research project "Abstract Models and Applications in Computer Science" no. 144017A, supported by the Ministry of Science, Republic of Serbia.

References

[1] Burke, B., Monson-Haefoel, R. Enterprise JavaBeans, 3.0. Sebastopol, CA: O'Reilly 2006.

[2] Chen, P., The Entity-Relationship Model - Toward a Unified View of Data. ACM Tensactions on Database Systems 1 br. 1 (1976), 9-36.

[3] Christof, E., Open Source Drives Innovation. IEEE Software 24 no. 3 (2007), 105-109.

[4] Daum, B., Professional Eclipse 3 for Java Developers. Chichester: John Wiley & Sons, 2005.

[5] DeMichiel, L., Keith, M., JSR 220: Enterprise JavaBeans, Version 3.0. Sun Microsystems. Santa Clara, CA, 2006.

[6] Fitzgerald, B., Kenny, T., Developing an Information Systems Infrastructure with Open Source Software. IEEE Software 21 br. 1 (2004), 50-55.

[7] Griffith, S., Richards, N., JBoss: A Developer's Notebook. Sebastopol, CA: O'Reilly 2005.

[8] Hanson, R., Tacy, A., GWT in Action. Greenwich, CT: Manning 2007.

[9] Keith, M., Schincariol, M., Pro EJB 3: Java Persistence API. Berkeley, CA: Apress 2006.

[10] Marrs, T., Davis, S., JBoss at Work: A Practical Guide. Sebastopol, CA: O'Reilly 2005.

[11] MaxDB. 2007. `http://mysql.com/products/maxdb/` (last accessed July 21.2007).

[12] Panda, D., Rahman, R., Lane, D. EJB 3 in Action. Greenwich, CT: Manning 2007.

[13] Sriganesh, R. P., Brose, G., Silverman, M. Mastering Enterprise JavaBeans 3.0. Indianapolis, IN: Wiley Publishing 2006.

Received by the editors July 30, 2007