Contemporary Java Web Technologies as a Service for the University Employees

Uroš Romić[1], Igor Manić[1], Ivan Pantelić[2]

[1]University of Belgrade, School of Electrical Engeneering; [2]Faculty of Business, Valjevo

Case study

DOI: **10.7251/JIT1202088R** UDC: **004.438JAVA:004.738.5**

Abstract: This paper describes the web application used by employees of the School of Electrical Engineering in Belgrade. The application is based on contemporary open source Java web technologies (including frameworks such as Spring, JSF, Hibernate, etc.). They were combined together into an advanced system that provides an environment for the rapid application development, high modularity and configurability. Paper describes main groups of application functionalities related to teaching process and financial operations, as well as additional functionalities. Application three-tier architecture is described in detail, with the description of technologies used in each tier. Application development environment is presented including build process management. Also, the security solution is described, as well as distributed computing model chosen for communication with other information systems within the School of Electrical Engineering.

Keywords: Java, Spring, JSF, Distributed Computing

Introduction

Employees of the School of Electrical Engineering (ETF) in Belgrade have been using web-based services for several years to access and input various types of information [1][2]. These services are currently implemented through application eZaposleni (e-Employees), which is a part of Integrated Information System (IIS) developed in the Computer Centre of School of Electrical Engineering in Belgrade, in cooperation with the development team of the Computer Centre of University of Belgrade (RCUB). It is a web application based on Java web technologies [3][4] and relies on two independent IIS subsystems – Faculty Information System (FIS), for monitoring and organizing the teaching process at the Faculty and FIMES, for personnel records, financial and material operations of the Faculty.

Separate instances of eZaposleni application, with Faculty-specific customizations, are used within IIS instances at about twenty faculties in Serbia and Bosnia and Herzegovina, including faculties at University of Belgrade, as well as Universities Singidunum and Sinergija. Since 2010, over 1000 user accounts were created, and about one million user requests were served.

Business Environment and Features Overview of Ezaposleni (e-Employees) Application

A significant part of the application functionalities depends on the interaction, within IIS, with previously mentioned FIS and FIMES systems. Therefore, two groups of functionalities which deal with the processes supported by each of these two systems should be pointed out and the most significant ones will be described below. Also, for certain functionalities, communication with information systems outside IIS is implemented based on different platforms, such as the faculty library information system and information system for technical support. The appli-

cation also includes a group of functionalities that are implemented as separate modules, independent of the surrounding systems, such as records of scientific papers and books, or the possibility of uploading and downloading documents.

Within eZaposleni application, functionalities related to the records of the teaching process at the faculty include: grades input, defining and entry of students' pre-examination and examination activities, input of data on the activities of teachers in the teaching process, search and review of students' data, overview of class schedules, examination and free hall schedules, overview of student surveys and others. Special attention was paid to functionalities for input and review of students' activities and evaluations, within which, free configuration of the number and scoring of pre-exam and test activities is provided. Thereby, support is provided for recording the data sets of points and grades, regardless of the manner in which exams of a particular subject are organized. Also, import and export of data of variable structure within .csv and .xls file is made possible, which further facilitated the process of teacher input and opportunity of automation of the process. In order to better organize grades input, teachers have the possibility of temporary or permanent transfer of permissions for input of grades and confirmation of other teachers' grade inputs. Functionalities for search and review of students' data provide the ability of searching by several groups of criteria, Cyrillic-Latin conversion of entered keywords, and an overview of the basic or broader set of students' data, depending on user privileges, which can be configured at each instance of application. An especially important place within the application, from a user point of view, hold functionalities related to input and review of the entered activities of teachers within the teaching process. Information about the held classes, exercises and laboratory exercises, as well as exam duties and review of exam assignments are recorded. The teacher enters and confirms completed obligations through the application features. The calculation of benefits on the basis of held educational activities is realized through mutual cooperation of FIS and FIMES systems. Within the groups of functionalities that are applied to the records of the teaching process, the functionalities intended for a specific group of privileged users, which are mostly Faculty authorities, have been separated. Functionalities intended for the authorities include review of engagement of employees in teaching, as well as generating reports, within which, there is a possibility of printing over 40 different reports related to the educational process (exam results and enrolment statistics, records related to classes, exams, surveys and so on).

Functionalities related to personnel records and financial and material business of faculty are partially or fully available to all users who are employed at the university, regardless of whether they are employed as teachers or administrative staff, and in accordance with their privileges. Among others, these functionalities include the review of salaries and fringe benefits to employees on various grounds, application for projects grants and overview of detailed information about them, review of worksheets, application of different email notifications by FIMES information system, possibility of sending user login to corresponding university departments of finance and personnel records, and others. Functionalities for review of salaries and benefits provide a detailed insight into the data on the payments, for the defined criteria related to the category, type of payment and time period. The data includes gross and net amounts, as well as information on tax and contributions for each payment. Records of projects are supported through the group of functionalities that allow a detailed review of project information for the managers of those projects, and the possibility of sending a request for opening projects to the Faculty authorities or the relevant department, by submitting detailed information about the project. As in the case of the educational process functionalities, in the case of personnel records and financial and material business there is a part of the functionalities intended for the Faculty authorities. Functionalities for the authorities include review of personnel information, review of the payment of salaries and compensation for employees, as well as an overview of projects and worksheets. Each of these functionalities includes review based on different criteria given by the user. Also, the Faculty authorities have the ability to review and update submitted applications for project opening, which automates the process of their approval.

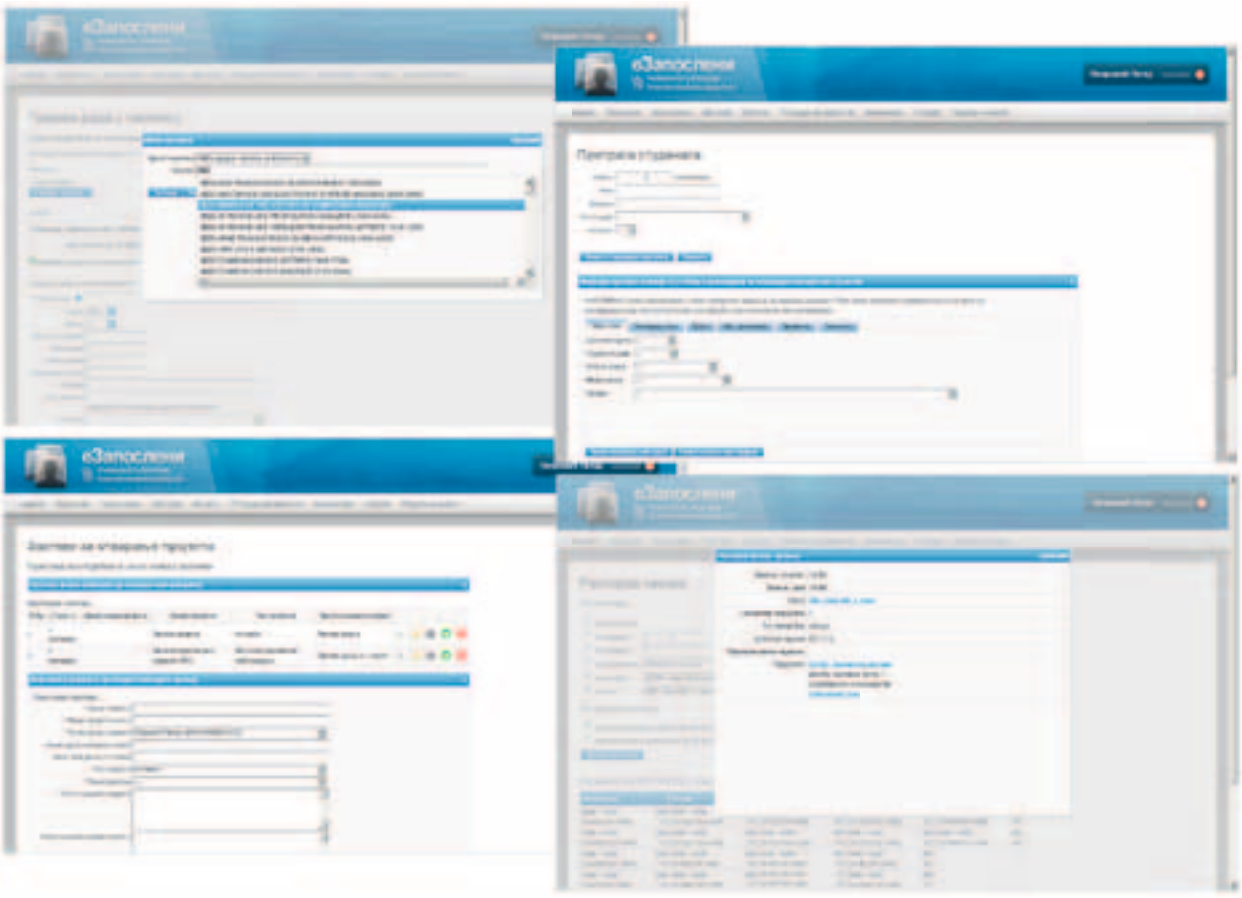

Figure 1 An Example of eZaposleni user interface.

Within a separate module, the functionalities for the records of published books and scientific papers published in international journals have been realized, which is the basis for establishing an up to date database of scientific research within the Faculty. Users have the option of input of papers published in journals from the Science Citation Index Expanded (SciE) list, and an overview of the entered papers for which they are registered as one of the authors. Once entered, paper is attached to all users of the application who are listed as its authors, thereby avoiding the need for multiple entries for each paper. This module includes a database with currently over 11,000 journals included in SciE list that have impact factors (IF) since year 2000. Information about the journal IF can be updated within special option, and there is a possibility of insight into the evaluation of papers based on the year of publication and the journal in which they were published. A special option for the privileged user enables export of predefined set of data on published papers, in XML, for the purpose of exchanging data on works with other systems. In a similar way as in the case of scientific papers, records of published books have been realized (textbooks, electronic textbooks, monographs, etc.).

In addition to these specific groups of functionalities that provide support for certain processes at the Faculty that require interaction with employees, eZaposleni application contains common functionalities, such as review of their own personal information and contacts, change of passwords, user manuals, as well as the ability to download different types of documents, forms and requests in electronic form. Setting and updating available documents is also possible for users with special privileges. System options are intended for system administrators and allow users to reset forgotten passwords and a detailed review of data on the use of application, in order to check the current load and the number of active users, and to prevent possible abuse.

Overview of Aplication Architecture and Used Technologies

The application is implemented using three-tier distribution architecture [5][6], in order to achieve modularity and ease parallel development and maintenance of the individual tiers. Three main tiers are Presentation, Application and Data tier, with Application and Data tiers being multi-tiered themselves, as described below.

Spring framework is used throughout the application, to manage and connect separate tiers, as it supports the integration of different technologies that were selected for this project [7]. Java classes that form the application's business logic are defined in separate XML files, as Spring beans. To define and manage dependencies between classes, Inversion of Control (IOC) principle is applied using Spring Dependency Injection (DI) mechanism. In this way, higher level of code re-usability and easier testing is achieved. One of the main benefits achieved by using Spring DI is the so called loose coupling, i.e. defining dependencies between objects and their behavior through the interfaces and defining the appropriate interface implementations via the meta-data. The implementation of particular interface may then be easily changed without changing the code, but only by changing the configuration files. Therefore, some of eZaposleni application tiers are divided into two packages, containing interfaces and implementation classes, in order to support loose coupling principle.

Data tier is responsible for integration with the RDBMS. Currently, PostgreSQL version 9 is used as the most advanced open source RDBMS solution. This tier is separated into Domain and DAO tiers, first one containing the Domain objects and latter containing DAO (Data Access Object) interfaces and implementations. Data is managed using Hibernate implementation of JPA specification. Domain objects are Java classes that map tables from relational databases using JPA annotations for object-relational mapping. DAO interfaces and implementations provide methods for manipulating data and include mostly simple CRUD operations. Hibernate uses entity manager object to maintain the integrity of the Domain objects.

Application tier provides business logic and it is divided into Service tier and Controller tier. Service tier methods provide implementation of application use cases, by communicating with one or more DAO tier interfaces. This layer is also responsible for the transaction management that is defined in a separate XML descriptor using Spring AOP, and is executed on the selected service methods. Spring AOP additionally simplifies the transaction management by defining aspects at meta-data level, and applying them transparently to the defined methods, at their execution time. In this way, a very important part of application business logic is clearly separated and can easily be maintained or changed. Controller tier consists of Java bean classes that are called controllers within application and are used as JSF managed bean, i.e. the communication between presentation tier and the controllers is implemented using JSF technology. Each controller supports user actions within one or more views using action methods that interact with one or more Service tier interfaces.

The third layer is the presentation layer that allows display of data to users. Graphical user interface is implemented using JSF 1.2 with RichFaces 3.3 component library to provide better interaction and more convenient use. Web pages conform to XHTML syntax, as a standard recommended by W3C (World Wide Web Consortium) [7]. RichFaces components have a built-in support for AJAX asynchronous calls to the server and allow partial loading of parts of web pages, which speeds up the GUI. Facelets technology is used for view rendering, and it also provides advanced page template mechanism in order to reduce development time and minimize the maintenance of GUI [8].

Spring Web Flow technology is used to navigate users between pages because it enables easy definition of complex scenarios of usage where it is necessary to guide users through the exact defined steps.

The modularity of the applications is further enhanced by the division into independent modules - packages that include groups of similar functionalities. It should be noted that presentation tier components, that is, JSF views, are also stored in these packages, as well as Spring XML declarations, making it easier to add or remove individual modules and export them into JAR libraries.

Spring Solutions Implemented in Ezaposleni Application

The "core" of eZaposleni application, as already said in previous chapter, is Spring framework (currently, version 3.0) and its DI mechanism [9]. DI represents pattern in which values of attributes of an object, during its creation, are placed from outside, by another object. In the case of Spring technology, the facility which implements DI is called Spring IoC Container, and is realized using two interfaces – BeanFactory and ApplicationContext, latter being the complete superset of the first and more frequently used. Information about classes that need to be instantiated, the extent of their mutual dependence is stored in the form of metadata, into XML configuration files. Instantiation of the IoC Container in the case of Java web application is done by adding the appropriate configuration (two context listeners) into web application descriptor (web.xml file).

In eZaposleni application, ApplicationContext interface was used to connect the dependent objects that are located in adjacent layers of the application. In addition of using the DI for connecting classes within the application, Spring framework's support for other tools was used and embedded, such as Hibernate or iBatis, which enables easier configuration of these tools. Again, DI templates and XML configuration was used, without additional Java code. Spring Aspect Oriented Programming (AOP) module is also used, primarily for managing transactions, as well as Spring Web Flow and partially Spring MVC (dispatcher servlet).

Important parts of the application structure are HTTP Invoker (used for implementation of distributed programming) and Spring Security (in the field of application security). The implementation of these technologies within eZaposleni application is described below.

Since it is a web application with a thin client, which should be available from anywhere on the Internet, special attention was devoted to the application security, that is, authentication and authorization of users. As a solution that meets all set requirements in terms of safety, as well as being a part of Spring technology, Spring Security project was elected. Configuring Spring Security is done by defining a corresponding class in XML files within the application, which is loaded by Application Context along with the other Spring XML definitions. Additional benefit, introduced in Spring Security from version 2.0, is so called Namespace configuration that allows easy setup of security options through the usage of predefined XML components. However, it must be noted that for each advanced configuration of security elements it is necessary to use the classic way of defining beans, which gives greater freedom in choosing classes for the realization of some functionalities. In eZaposleni application Namespace, configuration elements were used as much as they were able to respond to requests, while additional adjustments were made using standard definitions beans. Setup of the security options is separated in a special XML configuration file. Namespace configuration included the parameters definition of the basic <http> element, as well as its sub items - definition of login form, templates that determine which user roles would have access to certain URLs application, limiting the number of sessions per user etc. Additional settings, implemented by standard bean definitions, refer to the definition of appropriate classes for the connection with application database and fetching of users data (username, password, roles), as well as defining passwords encryption (SHA-256 with the username as a salt). There is a possibility for further extension of configuration by using classes that will decide on the possibility of access to the application on the basis of arbitrary parameters, such as IP address of the user.

It is mentioned that one of the most important requirements that was set is the efficient implementation of communication through the remote reference service with the existing and future versions of university information systems. Given the fact that eZaposleni and other surrounding systems rely on Spring technologies, for their communication, one of the following RPC models for which Spring libraries provide adequate support: RMI, Hessian / Burlap, HTTP Invoker or JAX-RPC/SOAP was elected [3]. HTTP Invoker model was chosen, which allows remote calls via HTTP and using standard Java serialization of facilities. HTTP communication is an important requirement, given the security constraints of the ETF network which would represent a problem for stan-

dard RMI services. Furthermore, Spring provides full support for the implementation of this model through the XML configurations. HTTP Invoker model is, on the "client" side of the application, implemented by declaring bean in the application service layer, which references the Spring class HttpInvokerProxyFactoryBean. Within the declaration, ServiceURL and ServiceInterface attributes were initialized. ServiceURL represents URL to which the application that provides the service responds. ServiceInterface is the name of the interface with service methods signatures. Declared HttpInvokerProxyFactoryBean has been included as an attribute into all other service beans which need fetching of data from remote systems. Within those beans it is treated as a standard Spring bean and there is no need for additional adjustments related to HTTP Invoker. On the "server" side, in an application that provides service, HTTP Invoker model has been realized through declaration of an HttpInvokerServiceExporter type bean, with service and serviceInterface attributes. The first attribute references the bean with service methods, and the second one the interface implemented by service bean. In order for the HTTP Invoker to translate customer requirements into service methods, it calls the corresponding servlet in web application descriptor that provides the service, which has been defined. Servlet name must be identical to the name of an HttpInvokerServiceExporter bean. Servlet URL pattern determines through which URL application the service will be provided, that is, which ServiceURL attribute will be used by HTTP Client Invoker.

Application Build Process and Development Environment

Application structure, build process, and dependencies are managed using Apache Maven 3. Multimodule application structure is defined in Maven's Project Object Model (POM) files, and further customization is realized using different parameters for each instance of application. In this way, creating application web archive (war) file for desired application instance is simplified and is performed by using single Maven command.

Eclipse IDE 3.7 (Indigo) is used as an application development environment, as well as Spring Tool Suite (STS), version 2.9.1 which is based on the Eclipse Indigo platform and comes with integrated features and pre-configured support for the development of J2EE applications based on Spring technology. Eclipse was chosen due to extensive configuration options, and as a good support in the form of supplements for other tools used for developing applications. Version control is implemented by using Subversion and Subclipse plug-in for Eclipse. Support for review and validation of Spring XML configuration files and bean definitions is provided by Spring IDE tools for Eclipse. In order to accelerate the development of GUI, JBoss Tools Eclipse add-on was chosen, which provides content assist, visual page editor, full compatibility with RichFaces and Ajax4JSF components, and support for both standard JSF and Facelets components. Apache Tomcat 7 is used as an application container in development and production environment. In production environment, Apache HTTP Server (httpd) is used in front of Tomcat, to provide additional security and configurability.

Conclusion

Solutions implemented in the application eZaposleni were chosen not only to facilitate communication and adjustment of surrounding systems, but also to be mutually compatible and, when possible, to effectively complement one another. Thereby, further development of the system in terms of adding new functionalities such as dynamic control of user privileges with the help of a Spring Security or multilingual support should be facilitated. Also, the transition to the new versions of used development tools and frameworks, or complete replacement of technologies in certain application layers is facilitated. Current development environment allows rapid adjustment of the system for different needs, so that it can be used as a template for a series of similar web applications. Finally, it should be noted that all used technologies are free of charge, which makes this system ideal for students' research projects.

Authorship statement
Author(s) confirms that the above named article is an original work, did not previously published or is currently under consideration for any other publication.

Conflicts of interest
We declare that we have no conflicts of interest.

References

[1] Web Portal for Employees, Computing Centre, School of Electrical Engineering, http://rc.etf.rs/projekti/veb_portal_za-posleni (in Serbian, Accessed: November 2012)

[2] Fu Xiaolong; Liu Qixin; Yuan Fang; Design and implementation of university level unified information system integration platform, 5th International Conference on Computer Sciences and Convergence Information Technology (ICCIT), 2010, Pages: 864 - 868

[3] C. Walls, R. Breidenbach (2008) Spring in Action, 2nd Ed, Manning Publications Co, Greenwich

[4] Wang HaiTao; Jia BaoXian (2010) Research Based on Web Development of Spring Integration Framework, International Forum on Information Technology and Applications (IFITA), Vol. 2, Pages: 325 – 328, 2010

[5] R. Hirschfeld (1997) Three-Tier Distribution Architecture, Washington University Tech. Report No. wucs-97-34

[6] Multitier architecture, http://en.wikipedia.org/wiki/Multitier_architecture (Accessed: November 2012)

[7] Arthur, J.; Azadegan, S.; Spring framework for rapid open source J2EE Web application development: a case study, Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, 2005 Pages 90 - 95 W3C, XHTML™ 1.1 - Module-based XHTML, http://www.w3.org/TR/xhtml11/ (Accessed: November 2012)

[8] B. Aranda, Z. Wadia (2008) Facelets Essentials: Guide to JavaServer Faces View Definition Framework, Apress, Inc., United States

[9] Rod Johnson, Juergen Hoeller, Keith Donald et al. (2009) Spring Java Application Framework Reference Documentation, http://www.springsource.org/documentation

Submitted: November 28, 2012.
Accepted: December 03, 2012.