Innovative Data Collecting System of Services Provided by Medical Laboratories

Adam Migodzinski, Robert Ritter, Marek Kaminski, Jakub Chlapinski, Bartosz Sakowicz

Abstract – **The article presents features of an innovative system that provides data collection of services provided by medical laboratories. System has been developed based on Java Enterprise Edition platform with usage of Spring and Hibernate frameworks combined with jQuery library.**

Keywords – **Spring Framework, Hibernate, Java, Java EE**

I. Introduction

Recently Internet has become the main source of information, entertainment, knowledge and platform for the rapid exchange of information. However, this is just one of the possibilities of using this powerful tool. In the last few years, strong growth of Internet's commercial use has marked. More and more companies began to share their databases of products and prices with the possibility to purchase them via Internet. A growing number of online shops caused creation of price comparison services - websites thanks to which users can quickly find an interesting product in the lowest price.

Presented application is innovative because there has not been introduced any website offering such set of services. Its introduction in future may be significantly easier for physicians and patients.

The aim of the work was to create a site collecting data on services provided by medical laboratories, with usage of open-source solutions (jQuery, Hibernate, MySQL, Tomcat) and technology based on Java EE and Spring framework [1,2].

II. Tools used to develop the system

Main idea of the project was to create a site, making use only of open-source libraries and projects. Authors decided to use Java and the Spring Framework as the foundation of the whole project, together with other supplementary technologies such as Hibernate or jQuery. Apache Tomcat has been chosen as application's server. System security has been assured through the use of Spring Security framework. Good knowledge of mentioned frameworks allows to accelerate the application development process. Unfortunately, their use does not guarantee success itself. Much depends on the programmer, who must remember to apply certain rules such as three-layer application architecture. Thanks to the program code becomes transparent and the development of applications in the future – easier.

Manuscript received November 09, 2011.
Katedra Mikroelektroniki I Technik Informatycznych
ul. Wolczanska 221/223 budynek B18, 90-924 Lodz, POLSKA
al. Politechniki 11, 90-924 Lodz, POLSKA
NIP 727-002-18-95
tel. +48 (42) 631 26 45 faks +48 (42) 636 03 27

III. System goals and functionality

The main aim of the project was to design system that would be a database of laboratories together with their offered medical examinations. Furthermore, it should provide quick searching of any examination with the possibility of price comparison. Such a system would constitute a huge convenience to both doctors and their patients searching for the best place to do the required tests [6].

System is addressed to various range of users. Due to this fact, division into four main roles of users was implemented. The roles are: laboratory worker, administrator, client and registered client. Each role has different functionality.

Laboratory worker role functionality:
- registration in the system
- adding examinations
- submitting newsletter content
- adding comments and files

Registered client can:
- search for examinations
- add opinion about laboratory
- register for newsletter

Client:
- search for examinations

Administrator functions are:
- moderating laboratory's opinions
- editing and sending newsletter
- placing commercial banners

IV. Architecture

Application has been designed in accordance with three-tier layer architecture (Fig. 1).

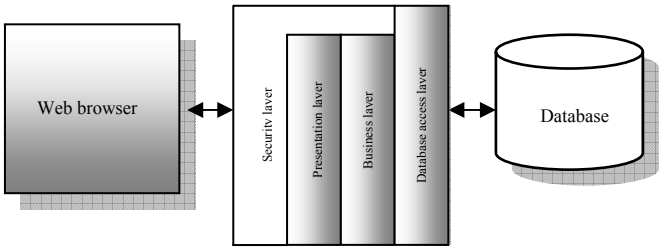

Fig. 1. Application's architecture

It distinguishes three independent modules. These modules are associated with each other by means of appropriate mechanisms to ensure communication between them and the

data transfer. The three modules are: presentation layer, business layer and database access layer. The correct model layer should be constructed so that the given layer uses the interface provided by the "lower" layer to communicate and have no knowledge of any "higher" layer. Such architecture is demanded by Spring Framework, which requires object-oriented programming with interfaces, loose-coupling between classes and modularity.

V. Business Logic layer

Business logic layer in the application has specific tasks. It collects data from a "lower" layer through its interfaces. Persistence layer forwards data to the logic layer as objects.
It is run by:

service-Java interfaces providing class' methods for implementing the service,

Java classes that define methods for implementing business logic depending on user requests. Responsible for the retrieval of data from layers responsible for the access to the database, saving new objects mapped to the appropriate records in the database, editing existing ones or deleting them.

VI. Persistence layer

Persistence layer is the lowest layer in the application [3,7]. It is responsible for retrieving data from a database using annotated POJO classes pursuing an object-relational mapping. It is implemented with:

– DAO interfaces – which share methods of classes to implement the DAO interface;

– Java classes that inherit from class HibernateDAOSupport, giving access to a wide range of methods for ease of use of data, such as adding to the database or erasing them, without worrying about releasing the session objects, transactions, or cleaning the cache memory. They operate on the class's entities;

– entities – POJO class with JPA annotations implements the object-relational mapping to the appropriate tables in the database.

This design does not require changes in source code after changing data persistence technology.

All data is stored in a MySQL 5.1 database. However, implementation of applications that run on a relational database in object-oriented programming languages such as Java can be time consuming and tedious. Facilitate and accelerate the action has been obtained by the usage of Hibernate, that is performing the mapping application skeleton representation of the object model of the relational model, using SQL. Hibernate's configuration is stored in XML file. There is defined connection through JDBC to the database and SQL dialect, so that system specific metadata can be generated. Example of Hibernate configuration is shown below:

```
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.dialect">
      org.hibernate.dialect.MySQLInnoDBDialect
    </property>
    <property
name="hibernate.connection.driver_class">
        com.mysql.jdbc.Driver
    </property>
    <property name="hibernate.connection.url">
      jdbc:mysql://localhost:3306/mediclabsdb
    </property>
    <property
name="hibernate.connection.username">root</pro
perty>
  </session-factory>
</hibernate-configuration>
```

In the project entities with annotations were used. Annotations in Hibernate are implemented in the Hibernate Core in the form of two independent packages: Hibernate Annotations and Hibernate EntityManager. Hibernate Annotations implements all annotations JPA / EJB 3.0. Java classes with annotations are replacing traditional XML mapping files. Below is presented Java class with annotations usage.

```
@Entity
@Table(name="authorities"
    ,catalog="mediclabsdb"
    , uniqueConstraints =
@UniqueConstraint(columnNames={"username",
"authority"})
)
public class Authorities  implements
java.io.Serializable {
     private Integer id;
     private Users users;

    public Authorities() {}
    public Authorities(Users users, String
authority) {
       this.users = users;
       this.authority = authority;
    }
    @Id @GeneratedValue(strategy=IDENTITY)
    @Column(name="id", unique=true,
nullable=false)
    public Integer getId() {
        return this.id;
    }
    public void setId(Integer id) {
        this.id = id;
    }
    @ManyToOne(fetch=FetchType.LAZY)
    @JoinColumn(name="username",
nullable=false)
    public Users getUsers() {
        return this.users;
    }
    public void setUsers(Users users) {
        this.users = users;
    }
```

VII. Presentation layer

Presentation layer is located at the top of three-tier architecture. It is responsible for implementing user's interface logic and contains the code navigating between web pages or displaying the forms. In presented application

presentation layer has been implemented in accordance with the MVC (model-view-controller) pattern, which includes:

– JSP pages – which are views responsible for presenting data to the user. Data is imported through the middle tier from database. Pages are operated by controllers;

– Controllers – Java classes that inherit from one of Controller class, depending on the kind of ongoing user request. Controllers communicate with the "lower" layer using the interfaces provided by it, import the required information and return the results to the appropriate view. One controller can support several views (Fig. 2).

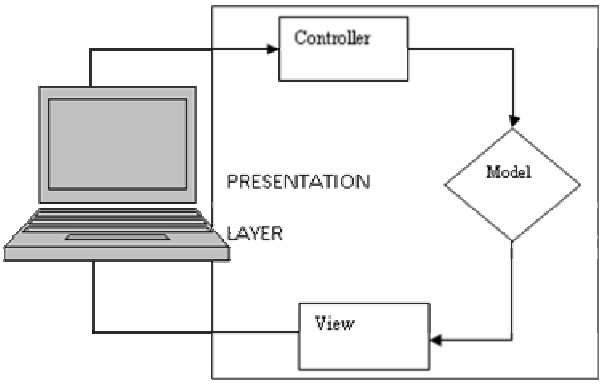

Fig. 2. Processing user's request step-by-step

Of course incoming request needs to be dispatched in some way. In other words it has to be known which controller is responsible of delivering essential data to JSP page. Spring provides several mapping methods but in presented project SimpleUrlHandleMapping was used. It maps controllers to URL adresses using a property collection defined in the Spring application context, as presented below:

```
<bean id="urlMapping"
class="org.springframework.web.servlet.handler
.SimpleUrlHandlerMapping">
   <property name="mappings">
      <props>
         <prop key="index*.htm">
              indexController
         </prop>
         <prop key="image/*.htm">
              imageController
          </prop>
       </props>
       </property>
```

</bean>

User interface has been enriched by jQuery plugins such as: tablesorter, masked input, autocomplete input field or ligthbox gallery [4]. jQuery is a cross-browser JavaScript library designed to simplify client-side HTML scripting. Implementing any plugin from those mentioned above is very easy. Basically it boils down to import appropriate plugin's script and putting path to it in <head> section. Next step is putting in separate JavaScript file methods that the plugin implements or extends and a single line "$(document).ready() executing specific actions. Sample usage of Autocomplete plugin is presented on Fig. 3 and the code is introduced below:

```
$(document).ready(function(){
$("input#cities").autocomplete({
source:[ "Zgierz", "Zgorzelec"]
    });
  });
```

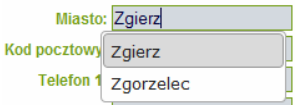

Fig. 3. jQuery UI autocomplete plugin in action

VIII. Security Layer

Ensuring application security is a critical aspect of its proper work [5,8,9]. When one needs to divide access to resources depending on user role, the help comes from the Spring Security framework. To work properly, "Spring Security" needs two tables to be created in the database: USERS and AUTHORITIES. First of these two must be fields storing two fields: username and password. In the second table must be username (which is a foreign key) and the name of the his role (authority). Spring Security configuration has been defined in a separate file - applicationContext-security.xml. It identifies the access to websites based on user role, the name of the page responsible for logging in, redirects to the appropriate page when one logs on, logs off or if the login fails:

```
<http auto-config="true" lowercase-
comparisons="false" access-denied-
page="/index.jsp">
<intercept-url pattern="/login.jsp"
access="IS_AUTHENTICATED_ANONYMOUSLY" />
<intercept-url pattern="/index*.jsp"
access="IS_AUTHENTICATED_ANONYMOUSLY"/>
       ...
<intercept-url pattern="/profile.jsp"
access="ROLE_USER, ROLE_ADMIN" />
<form-login login-page="/login.jsp"
authentication-failure
url="/login.jsp?login_error=1" />
<logout logout-url="/logout" logout-success-
url="/index.jsp" />
</http>
```

To enable the security methods for applications, filers capturing users requests need to be configured in the application descriptor (web.xml file), as shown below:

```
<filter>
        <filter-name>springSecurityFilterChain
         </filter-name>
        <filter-class>
org.springframework.web.filter.DelegatingFilte
rProxy
         </filter-class>
</filter>
       <filter-mapping>
        <filter-name>springSecurityFilterChain
```

```
        </filter-name>
        <url-pattern>/*</url-pattern>
</filter-mapping>
```

IX. USAGE OF JMS AND CKEDITOR

The system introduces the ability to send newsletters to users who have expressed their desire to receive it. Spring Framework has an abstract API that makes sending e-mails a relatively simple process. A main element of that API is an interface MailSender, which has two different implementations. In the project JavaMailSenderImpl was used. The reason why this one has been chosen is its possibility of sending the MIME messages. Responsible for sending emails is method sendEmail, located in the class EmailServiceImpl class. This method creates and send messages to each customer. For the proper work of the mechanism, relevant beans must have been defined in the applicationContext.xml file:

```
<bean id="mailsender"
class="org.springframework.mail.javamail.JavaM
ailSenderImpl">
     <property name="host">
<value>${host}</value></property>
     <property name="port">
<value>${port}</value></property>
     <property name="username">
<value>${username}</value></property>
     <property name="password">
<value>${password}</value></property>
     <property name="javaMailProperties">
       <props>
      <prop key="mail.smtp.auth">true</prop>
       <prop
key="mail.smtp.starttls.enable">true</prop>
           </props>
     </property>
</bean>
```

PropertyPlaceholderConfigurer loads properties from one or more external property files and uses those properties to fill in placeholder variables in the bean wiring XML file.

```
<bean id="propertyPlaceholder"
class="org.springframework.beans.factory.confi
g.PropertyPlaceholderConfigurer">
<property name="location"
value="mailsystem.properties"></property>
</bean>
```

Fig. 4. Properties import from external property file

X. CONCLUSIONS

In recent years much has changed in approach of creating applications that run on the web server. Role of frameworks, which support creation of application, its development and testing has increased. Examples are the Spring Framework (for Java), Code Igniter (PHP). NET Framework and many others. The aim of this study was to establish a system for collecting information of medical laboratories and their services. It would greatly facilitate the work of doctors and saved patient's time who is searching for relevant laboratory to do the examination. Such a system could improve the quality of services due to the possibility of comparing prices or adding an opinion of the laboratory.

ACKNOWLEDGEMENTS

The authors are a scholarship holders of project entitled "Innovative education ..." supported by European Social Fund.

REFERENCES

[1] Craig Walls, Ryan Breidenbach, „Spring in Action – Second Edition", Manning Publications, 2007, ISBN: 1-9339-8813-4.

[2] R. Johnson, J. Hoeller, A. Arendsen, T. Risberg, C. Sampaleanu, „Professional Java Development with the Spring Framework, John Wiley & Sons, 2005, ISBN: 0-7645-7483-3.

[3] Christian Bauer, Gavin King, "Hibernate w akcji", Helion, 2007, ISBN: 978-83-246-0527-9.

[4] Bear Bibeault, Yehuda Katz, "jQuery in Action", Manning Publications, 2008, ISBN: 1-9339-8835-5.

[5] John Arthur, Shiva Azadegan, "Spring Framework for Rapid Open Source J2EE Web Application Development: A Case Study," snpd-sawn, pp.90-95, Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Networks (SNPD/SAWN'05), 2005.

[6] Niziałek, A.; Zabierowski, W.; Napieralski, A.: "Application of JEE 5 technologies for a system to support dental clinic management"; Modern Problems of Radio Engineering, Telecommunications and Computer Science, 2008 , ISBN978-966-553-678-9

[7] Ziemniak Piotr, Sakowicz Bartosz, Napieralski Andrzej: "Object oriented application cooperation methods with relational database (ORM) based on J2EE Technology"; CADSM'2007; 9th International Conference The Experience Of Designing And Application Of Cad Systems In Microelectronics, Polyana, Ukraina, 20 - 24 February 2007, str. 327-330, s.593, A4, ISBN 978-966-553-587-4, wyd. Publishing House of Lviv Polytechnic National University 2007.

[8] PilichowskiI M., Sakowicz B., Chłapiński J.; "Real-time Auction Service Application Based on Frameworks Available for J2EE Platform", pp. 166-169, Proceedings of the Xth International Conference TCSET'2010, "Modern Problems of Radio Engineering, Telecommunications and Computer Science", Lviv-Slavsko, Ukraina, 23-27 February 2010, s.380, A4, wyd. Publishing House of Lviv Polytechnic National University 2010, ISBN 978-966-553-875-2

[9] Marcin Mela, Bartosz Sakowicz, Jakub Chlapinski: "Advertising Service Based on Spring Framework", 9th International Conference Modern Problems of Radio Engineering, Telecommunications and Computer Science, TCSET'2008, 19-23 February 2008, Lviv-Slavsko, Ukraine, ISBN 978-966-553-678-9