Annales UMCS Informatica AI 6 (2007) 161-170

Annales UMCS
Informatica
Lublin-Polonia
Sectio AI

http://www.annales.umcs.lublin.pl/

Application elasticity as the factor determining the choice of technology. Testing of the factors determining the choice of application technology

Barbara Gocłowska, Zdzisław Łojewski*

Institute of Computer Science, Maria-Curie Skłodowska University,
pl. M. Curie-Skłodowskiej 1, 20-031 Lublin, Poland

Abstract

The learning creation interface constituting one of the most important elements incorporated in the Adaptive System LISE has been discussed in the present paper. The technologies associated with the use of JSF and EJB, Swing as well as GUI Builder have been selected for the testing.

JDBO and JDO packages have been used in order to compare various technologies of connection with the database. The tests have been carried out on the MySQL server and Sun Java System Application was used as the application server. The obtained results enable the choice of optimal solution.

1. Introduction

In order to create an adaptive learning system, it is necessary to prepare tools enabling creation of the course content. Owing to a limited number of course authors, on-line accessibility of those tools is unnecessary. Therefore an applet seemed to be the simplest method to prepare the program performing the author interface function. Its task would consist in the edition, updating and creation of new courses, chapters, units, quizes etc. Those entities would be recorded directly in the database, for instance MySQL 5.0 or OracleXE[1]). Owing to a wide spectrum of available technologies, thorough analysis of their advantages and disadvantages is extremely important. The technologies associated with the use of GUI Builder and swing package have been selected for testing. Such a type of solutions was compared with the Java Server Faces and Enterprise Java Bean technologies. After the analysis, we decided to select the most elastic technology to be used for developing of the course content creation package.

*Corresponding author: *e-mail address*: loe@tytan.umcs.lublin.pl

[1]We used both database servers in order to perform the tests but not to distribute the data.

The use of the components incorporated in the TopLink non-standard package has been also discussed in the present paper. That package as well as the standardized TopLink enable the query using the relational databases in an object – oriented manner. In order to compare the above mentioned technologies with Java Database Connection, the learning creation interface in the Adaptive System LISE [1] was prepared using the above mentioned connection elements.

Advantages and disadvantages of those solutions are discussed in the present paper. The elasticity of their use i.e. further potential extension, upgrading of functionality etc. were the determinant factors for the selection. The commonly known and generally used design standards were used in the implementations described herein, among others: Model, Viewer, Controller and Strategy [2].

The analysis was illustrated by means of sequential diagrams being the best modelling tools to reflect the flows of information between the individual application layers and within the objects being created.

All tests were performed on the MySQL[3] server, using the Sun Java System Application[4] as the application server.

1. The applets created in the swing and with the use of GUI Builder

Applet has been performed by means of two methods. The first, easier method consists in the use of GUI Builder. NetBeans 5.5 environment being used as our support when any applications are created, is characterized by extended tools for backing of finished components and their adding to the panel being created actually on the forms. Obviously, java bean components, for example those which are responsible for the connection with data base or associated with the creation of table model, exceptions etc., are also needed. Those components can be repeatedly used (which was obviously done by us) in the application created by means of different methods. The use of GUI Builder enables immediate adding of components, their easy arrangement as well as semi-automatic creation of programming of events associated with the components. Furthermore, GUI Builder is characterized by a special „programmer friendly" feature enabling adding of java bean components directly to the panel by means of drug and play method and their backing as the components with all resulting consequences.

Similarly to the case of UI components, created for JSP or JSF website, for instance a component backed in that manner and incorporated in the JDBCRowSet class object added by us, enables unrestricted choice of the databases server controller, isolation level, the owner or databases users etc., irrespective of the fact that DatabaseMetaData interface has not been used by us for the JDBC object definition [5].

However, besides its advantages, such solution is characterized by some disadvantages. A part of code is generated by the environment and provided in the layout generally differing from that to which the program author is accustomed. Any direct intervention into certain code fragments or addition of the comments etc. is impossible. Obviously that problem can be easily resolved by means of some changes, for instance from the simple editor level in the form of Notebook. However, in our opinion, the major problem is the excessive extension of individual classes. For instance, more than 3000 rows are included in the created class AuthorEditor. The great efforts are required for searching for the declarations and methods distributed therein. The application created directly in Swing is free from such in disadvantage. Additionally, the maintenance of the program is extremely easy owing to the distribution of individual elements (e.g. the creation of individual class objects for the majority of panels and dialogue boxes). The class objects enable multiple use after insignificant modifications. Their essential feature is easy testing possibility for individual classes.

The problem associated with the access to various database controllers and first of all with the possibility of easy download of various database tables, which is extremely important for the program with the main purpose consisting in insertion, updating and creation of new entities to the database, was resolved by us by means of the above mentioned interface defining the operating methods for metadata. The use of the DataBaseMetaData component in the course of action programming of the button used to search the table with specified name from pull-down list has been illustrated below in the form of code fragment. The elegant and perfect switching between the database tables, without necessity to create the individual panel for every table should be emphasized.

```
private void nextButtonActionPerformed(java.awt.event.ActionEvent evt) {
        try{
            Class.forName("org.gjt.mm.mysql.Driver");

connection=DriverManager.getConnection("jdbc:mysql://localhost/tutor",
username, password);

            statement = connection.createStatement();
            databaseMetaData = connection.getMetaData();
            ResultSet metaDataResultSet =
databaseMetaData.getTables(null, null, null, new String[] {"TABLE"});
            while (metaDataResultSet.next())
                  tableNames.addItem(metaDataResultSet.getString(3));
            metaDataResultSet.close();
         } catch (Exception ex) {
```

```
            JOptionPane.showMessageDialog(this, ex);
        }
    }
```

It appeared to be very convenient in practice (first of all for the program user, but also for the programmer, because the creation of individual classes for every table is unnecessary).

However, irrespective of perfect transparency of the code for the program created in Swing, its easy maintenance owing to subdivision into individual „modules" (Fig. 1), air-tightness etc. the advantages of such solution can not be compensated owing to the efforts required to arrange the elements in the panel and to achieve an attractive appearance.

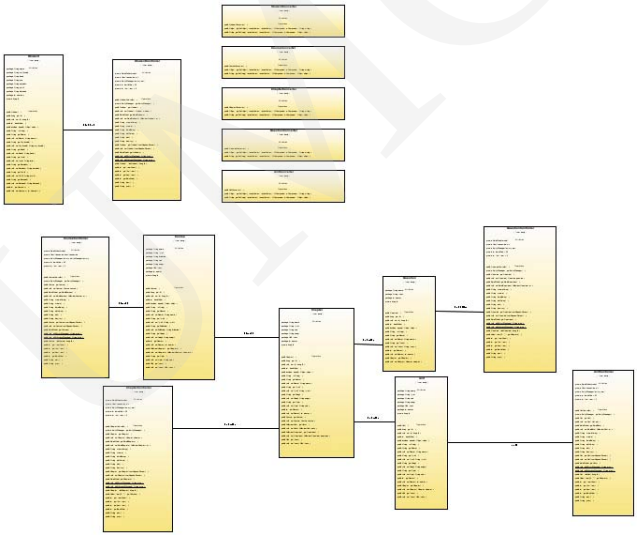

Fig. 1. Diagram for classes of the program InterfejsAutora created in Swing

In both types of the programs described above, the process of connection between GUI and database is possible using the object in the DriverManager or DataSource class. The second solution is more convenient in practice and recommended by the programmers employed in the Sun company. Both programs can be used as independent applications or displayed in the search engine window.

2. Application in Java Server Faces technology

The next version of the learning courses editor creation consisted in the creation of a web application. Therefore we used the Java Server Faces technology enabling application of various access layers. One of the first

problems to be resolved was our intention to obtain the multiple use components, similar to the possibilities available in the case of GUI Builder.

Using the JSF technology for data presentation, we are able to create the UI component libraries very quickly. We used this opportunity, e.g. implementing the logon component included in many applications and becoming a typical multiple use component. Let us refer to Figure 2 for the representation of a backed JSF component for which the classes TagHandler, TagLibrary.tld, as well as BeanInfo were programmed as illustrated in listing the fragments included below.

Fig. 2. Component backed in JSF, recorded in tld. library

The definition of the library (LogonDescriptor.tld) (Listing 1) and the recording of tags programmed for creation of UI elements therein was necessary for the component backing [6].

The library defined in such a manner should be declared on the view page in order to enable its use.

```
<?xml version="1.0" encoding="UTF-8"?>
<taglib version="2.0" xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee web-
jsptaglibrary_2_0.xsd">
   <tlib-version>1.0</tlib-version>
   <short-name>nowydeskryptor</short-name>
   <uri>/WEB-INF/tlds/NowyDeskryptor</uri>

   <tag-file>
      <name>PlikTagowDeskryptora</name>
      <path>/WEB-INF/tags/PlikTagowDeskryptora.tag</path>
   </tag-file>
   <tag-file>
      <name>EleganckiLogowacz</name>
```

```
    <path>/WEB-INF/tags/EleganckiLogowacz.tag</path>
  </tag-file>
  <tag-file>
    <name>LogowaczElegant</name>
    <path>/WEB-INF/tags/LogowaczElegant.tag</path>
  </tag-file>
</taglib>
```

Listing 1. Descriptor for tags coding graphical components for the logon page

```
<%@tag description="Plik tagów deskryptora" pageEncoding="UTF-8"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@taglib prefix="f" uri="http://java.sun.com/jsf/core"%>
<%@taglib prefix="h" uri="http://java.sun.com/jsf/html"%>
<%@attribute name="logowaczElegant" required="true"%>
```

Listing 2. Definition of component attributes for the logon window

```
<%@ taglib prefix="umcs" tagdir="/WEB-INF/tags" %>
```

Listing 3. Address of library used in the application view pages

The objects from the classes responsible for connection with the database created for the application in the Swing technology were used again in the JSF application. The components for view pages management (one component for every JSP web) and one common component have been added. The corresponding declarations for managing components were introduced in the file *web.xm*l and navigation rules (mainly for dynamic rules) as well as navigation logics code were defined and incorporated in the components managing *javaBean*.

The effect appeared to be very positive, because extraordinary code transparency was achieved with insignificant effort. Additionally, the possibility to perform the records regarding the manner of functioning and operation range for the components JavaBean in the layout descriptor `faces-config.xml` as well as the manner of navigation, increase the application transparency. In our opinion, the access to the program by means of search engine should not be interpreted as the advantage of that specific program, but as a right solution in respect of its use on the long term scale.

3. JSF technology in connection with EJB components

Any object – oriented query using the relational database is possible by means of the Enterprise Java Bean technology. The use of entity type

components managed by means of container appeared to be the most convenient communication method in practice.

The possibilities accessible through the layer described in the separate Java Persistance specification are extremely extended, but transparent. The generation of huge fragments of codes responsible for download and updating of database, and also the creation of new tables using the „in flight" method is possible by means of non-standard implementation of POJO introduced by the Sun company under the name TopLink [7].

It is extremely convenient in practice. The created code is extremely transparent and easy to maintain owing to modular character of the objects being created. Another aspect of JDO use in the NetBeans 5.5 environment i.e. the generation of the view pages enabling immediate testing of the application, should be also emphasized. The task of the programmer consists in the adaptation of view pages and functionality to the objectives pre-assumed by the programmer.

Furthermore, the application is characterized by the greatest advantage of EJB technology i.e. portability. The tables of database being created are included in the design folder, as illustrated in Figure 3, and the end user is really independent of the database server.

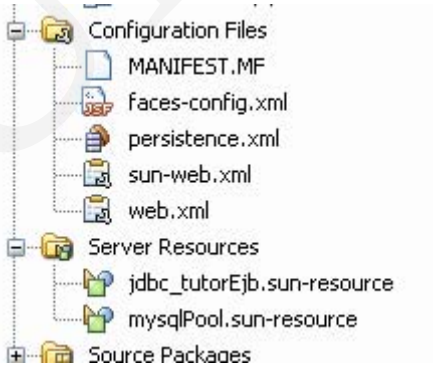

Fig. 3. Fragment of Editor author design folder created in the JSF and EJB technologies

As in the case of the program created in the JSF and Java technologies described above, we used the javaBean components managing the view page. However, in this case they are generated by the programming environment. Their default listing (Listing 4) illustrates the fragments of descriptors used to define the operation range of the components managing the view pages.

```
<h:commandLink action="#{course.createSetup}" value="Utwórz nowy
                                        kurs"/>

<managed-bean>
```

```
        <managed-bean-name>course</managed-bean-name
        <managed-bean-class>encje.CourseController</managed-bean-class>
        <managed-bean-scope>session</managed-bean-scope>
</managed-bean>

public String createSetup() {
        this.course = new Course();
        return "course_create";
    }

<navigation-rule>
        <navigation-case>
            <from-outcome>course_create</from-outcome>
            <to-view-id>/course/New.jsp</to-view-id>
        </navigation-case>
    </navigation-rule>
```

Listing 4. The component method CourseController – course.createSetup is called out in the page course.List.jsp. The method reverses the navigation rule "course_create" redirecting to the website course.new.jsp

The operation algorithm for dynamic websites navigation is similar to that described for the first Internet application, as illustrated in Figure 4.

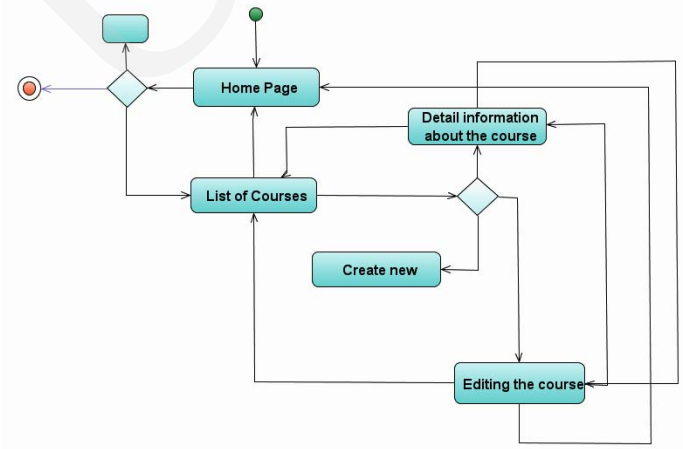

Fig. 4. Algorithm for websites navigation, recorded in the files javaBean, config-sys, and in view pages jsp

The classes EntityManager and EntityMenagerFactory [8] are not responsible for the connection with the database in this case. The comparison of the use of JDBO and JDO [9] interfaces and classes is illustrated in Fig. 5.

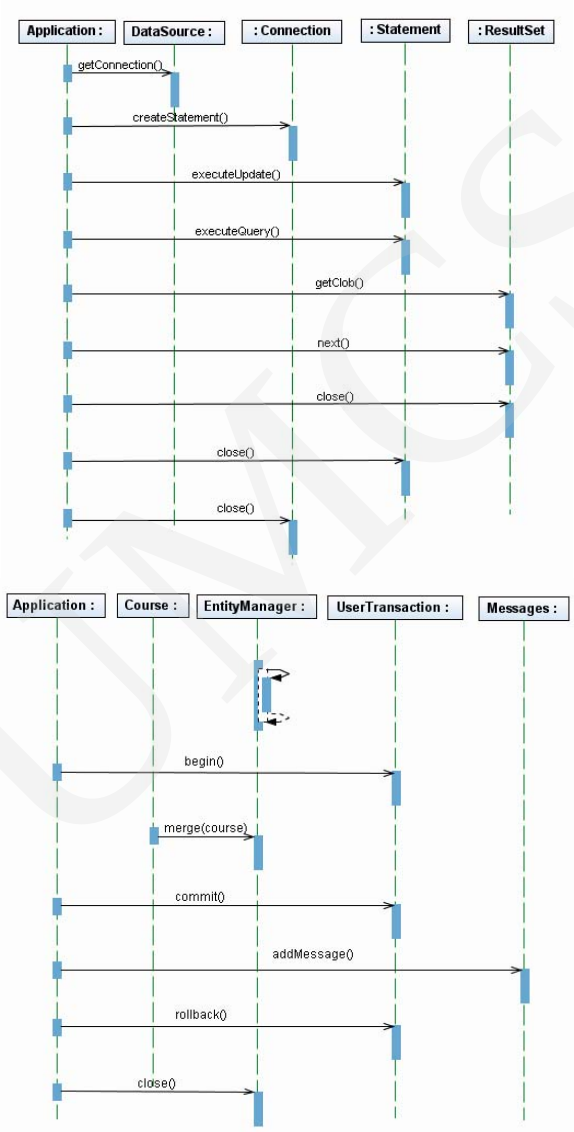

Fig. 5. Diagram for sequence of connection with database for JDBO technology and JDO in RH column

Summary

In order to select technology in the course of creating the program whose purpose is to enable easy data handling for the courses author as well as convenient and quick operation for the programmer, we analysed several

variants. A Stand-alone application created using Swing packages, GUI Builder, JSF technology and JDBC or JDO optionally was tested in order to answer the question associated with the choice of an optimal technology.

The programming by means of Swing package appeared to be labour consuming and rather inefficient. Using GUI Builder we were able to create elegant windows, but it was in conflict with the object – oriented programming idea in Java. From our point of view, the combining of JSF technology and entity components (EJB 3.0 components) is absolutely the most effective solution. The tests also demonstrated that non-standard solutions JDO (Java Data Object) supplied by Sun[], known as TopLink mark were extremely useful and in our opinion more convenient in use in comparison with other solutions available at the moment. Surprisingly positive results were obtained by combination of JSF and JDO technologies. Furthermore, the obtained application is characterized by some unquestionable advantages i.e. independence of the system and database server, quick software creation and first of all the subdivision of the application into several layers.

References

[1] Gocłowska B., Łojewski Z., *Intelligent Tutorial System LISE*. Annales UMCS, Informatica AI, 5 (2006) 441
[2] Shavor S, Fairbrother S., Kehn J., McCarthy P., *Java Developer's Guide to Eclipse*. polish edition Helion, Gliwice, (2005) 782.
[3] MySQL 5.0, Reference Manual, Copyright MYSQL AB, http://dev.mysql.com/doc/refman/5.0/en/index.html 16.02.2007,
[4] Jendrock E., Ball J., Carson D., Evans I., Fordin S., Haase K., The Java™ EE 5 Tutorial For Sun Java System Application Server Platform Edition 9, (2006).
[5] Horstmann C., Cornell G., Core Java 2, *Advanced Features*. Polish edition Helion, Gliwice, (2005) 1143.
[6] Goei E., Stearns B., *Writing Custom Components for Java Studio Creator Part 2: Design-Time Considerations*. 16.02.2007, http://developers.sun.com/prodtech/javatools/jscreator/reference/techart/2/writing_components_rules.html
[7] Understanding Oracle TopLink, 16.02.2007, http://www.oracle.com/technology/products/ias/toplink/doc/10131/main/_html/undtl.htm
[8] Tyagi S., McCammon K., VorburgerM., Bobzin H., *Java Data Objects*. Polish edition Helion, Gliwice, (2004) 456.
[9] Monnox A., *Rapid J2EE Development: an Adaptative Foundation for Enterprise Application*. Polish edition, Helion, Gliwice, (2006) 478.