Sead Mašović[1)]
Muzafer Saračević[1)]
Hamza Kamberović[1)]
Esad Međedović[2)]
Faruk Selimović[3)]

1) Prirodno-matematički fakultet u Nišu, {sekinp, muzafers, hamzanp}@gmail.com

2) Tehnički fakultet u Čačku, e-mail: esad@dr.com

3) Univerzitet u Novom Pazaru-Departman za prirodno-tehničke nauke

INTEGRATED MANAGEMENT WEB-BASED APPLICATIONS ON JAVA PLATFORM

Abstract: *The appearance of the Internet has led to a revolution in many spheres of human activity, however, probably the biggest impact it has had in developing the global business environment. Information, their availability and speed of exchange are the basis of successful modern business enterprise, and the Internet in this context is the perfect tool for finding and distributing information. Complex, Web applications have become essential for survival and development of modern enterprises in a global, international markets. Web applications are by nature distributed applications, which means that they are programs that run on more than one computer and communicate via a network. Specifically, the web application is accessed via a web browser and therefore update and maintain web applications without installing special software on potentially thousands of client computers is a key reason for their popularity. This paper describes integrated management and the production of Web applications in the Java programming language.*
Keywords: *Internet, Java, Java Servlet, JavaServerPages.*

1. INTRODUCTION

This paper shows the development of Web applications in the Java programming language. Web applications are used in order to respond to the users' requests, i.e. generate Web pages on the client's request. The applications are stored on the server and operate on the principle of client-server connection, i.e. client sends a server request, when the server fulfills the request, it sends a response to the client. To create Java applications, Servlet and JSP technology were used. Java Servlets are Java classes that extend the functionality of the server. They are executed on a Web server and act as a connection between the client and the server and their life cycle lasts for as long as there is a connection between the client and the server. JSP technology is a mixture of HTML and Java. It is used to display text on the Web, but has full functionality of Java.

The application is created in the three-layer architecture. It is divided into three layers - presentation, business and data layer. Presentation layer contains HTML or JSP files and relies on the business layer that takes care of business logic, rather the data-handling. Business layer is built in Java Servlet technology. The data layer stores data that can be retrieved or changed by the business layer. Three-layer

architecture is used because the applications built into it are easy to transfer and layers are independent. If you need to switch applications from MS Windows operating system onto the Linux operating system, it will not be necessary to change the whole application but only a part of it. Three-layer architecture is similar to the MVC architecture, Model - View - Controller. The difference is that the MVC architecture doesn't have completely separate parts as in the layer architecture. In MVC architecture, some parts contain business logic among the other things. The paper also gives an example of a simple Web application designed using three-layer architecture. The application allows you to enter data into the student database.

2. JAVA TECHNOLOGIES

In one of the first definition of "Sun" company, Java is described as follows:

Java is: a simple, object-oriented, distributive, interpreted, robust, secure, architecturally neutral, portable, of high performance, multithreaded and dynamic program language.

Java is completely independent from the operating system on which it runs, which insures transferability (portability) of applications developed in Java. Written applications can be executed on any operating system, assuming that the operating system has JRE (Java Runtime Environment) or JVM (Java Virtual Machine) that interprets Java instructions, turning them into specific operating system orders. Also, the programming language is completely independent from the specific database management systems (DBMS - Database Management System), through which, with providing the appropriate drivers, connection of applications to the specific DBMS systems and mapping general Java commands in the order specific DBMS is enabled.

We distinguish three editions of the Java platform. Each edition is intended for the development of specific applications and provide different execution (runtime) environment:

1. Java 2 Standard Edition, J2SE - provides execution environment for the development of desktop applications and represents the core of functionality for the next editions of the Java platform;
2. Java Enterprise Edition, JEE - a superset of J2SE, which supports the development of enterprise applications;
3. Java 2 Micro Edition, J2ME - defines a set of execution environments for development of applications for embedded devices such as mobile phones, PDAs, TVs and other devices that have a lack of resources to support J2SE or JEE.

2.1. WWW and Java as a platform for clients

Wide acceptance of the Internet or World Wide Web, has produced, among other things, a very important effect: the Web browser interface (browser) has become familiar to most PC users. Information systems in which the communication takes place with the user through a Web browser, eliminates to a large extent the need for long-term and expensive training.

On the other hand, Java is a programming language whose major characteristic is a complete independence from the physical platform on the level of translated code. Java programs, in the form of applets, can be embedded in Web pages and thus distributed to users. The consequence of this is the possibility of automatic distribution and installation of client applications on the network, regardless of the client's actual physical platform - Web browser with support for Java is enough for this.

The combination of WWW and Java technology has enabled the implementation of client/server information systems which, unlike

traditional systems, are characterized with the following features:

- simple and widely accepted form of user interface (Web browser);
- automatic distribution and installation of client applications;
- simpler maintenance, especially in heterogeneous networks.

2.2. Development of Web applications in the Java programming language

Web applications are applications that are created on request by dynamic web pages, and shall be executed on the platform provided by the Web server. How does a Web application work? The client connects to the server over IP (Internet Protocol) address and sends some request. Depending on the request, server generates a response and sends it to the client. The client is actually a Web browser (e.g. Internet Explorer, Mozilla Firefox, Google Chrome, Opera) responsible for the correct display of responses. Web applications in the Java programming language can be implemented using two technologies: Java Servlets and Java Server Pages (JSP). [1, 2] The optimal result is achieved by a combination of both technologies, as described below.

2.3. Java Servlet Technology

Servlets are Java objects that dynamically process requests and generate responses. Generated response is usually in the form of HTML (HyperText Markup Language), but the data can be in another form, for example XML (Extensible Markup Language). [3, 4] Servlets run on the server and the program code is never disclosed outside the server, only derived results are submitted. This way, the author code is fully protected, as opposed to applications that run in a Web browser using the Java Applet technology. [5] Another more advantage of using Servlets is their portability. They can run on different servers without changing the code. [6]

How does a Servlet actually work? After the client sends a request to a server, it loads the servlet and creates a separate thread. Each servlet has the methods *doGet()* or *doPost()*, which are used to process the request. For each request a separate thread is created, which makes execution significantly faster than any other technology, in which, operating system must create a new process for each request, initialize it, run it and clean up after it, which is extremely slow. One example of such technology is the CGI technology (Common Gateway Interface).

Each Web servlet takes over the class *javax.servlet.http.HttpServlet*, and has defined methods of *doGet()* or *doPost()* which accept the argument *HttpServletRequest* and *HttpServletResponse*. *HttpServletRequest* class contains methods by which we can find out information sent by users, while *HttpServletResponse* class has methods by which data is sent to the user. Simple Web servlets use *PrintWriter* class for sending content to the user. GET requests transfer parameters through URL (Uniform Resource Locator), which is, due its limited length, inappropriate for transferring a larger number of parameters. POST requests transmit parameters in the body of the request, regardless of the URL, which is perfect for sending a large number of parameters or even the entire database.

An example of a simple servlet is shown in the code segment 1. Using *PrintWriter* class, HTML document can be directly generated. The downside of this approach is that a large part of the servlet output consists of HTML tags. The user interface is included in the servlet, which means that when changing the user interface all servlets must change and re-translate. [6]

Code segment 1. An example of a simple servlet

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloServlet extends HttpServlet {
public void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.println("<html">);
out.println("<head><title>Hello World</title></head>");
out.println("<body>Hello World!</body>");
out.println("</html>");
out.flush();
```

2.4. JavaServer Pages (JSP)

JSP technology (JavaServer Pages) is also a technology that allows generation of dynamic Web pages. However, unlike the Java Servlet technology, parts of Java code are inserted in the static content. [7] JSP technology works so that it places tags similar to HTML tags in the JSP files that end with the extension *.jsp*. These files are similar to HTML files, but differs from them by JSP tags *'<%'* and *'%>'* in which the Java code is embedded. That code is on s request of a Web page translated into servlet. The process of translation is done by a server. When a user requests a Web page, Web server checks whether the *.jsp* file is newer than *.class* file associated with it. If the *.jsp* is newer or *.class* file does not exist, the Web server translates *.jsp* file before it runs it. Until *.jsp* file does not change, the Web server does not have to re-translate the class. [6]

An example of *.jsp* file is shown in the code segment 2. The program receives a number as a parameter and prints its square.

JSP code elements and their syntax are listed in the table 1. Directives are not used to generate the output. Instead, they "teach" JSP translator how to translate the code.

Code segment 2. An example of .jsp file

```
<%@ page language="java" session="false" contentType="text/html; charset=UTF-8" %>
<html>
<head><title> Squares of numbers</title></head>
<body><table cols="2" border="1">
<tr><td><b>Number</b></td><td><b>Squares of numbers</b></td></tr>
<%
String sNumber = (String)request.getParameter("number");
Integer no = null;
if(sNumber!=null) {
try {
no = Integer.valueOf(sNumber);
} catch(Exception ex) {}
}
out.print("<tr><td>"+no+"</td><td>"+(no.intValue()*no.intValue())+"</td></tr>");
%>
</table></body></html>
```

For example, lines of code

*<%@ Page import = "java.io. *"%>*

means that you should install the package *java.io*. Declarations serve to define global variables and methods. Listing, in addition to using *out.print()*, is possible to achieve by labels *'<%='* and *'%>'*.

Table 1 JSP directive and scripts' elements[6]

Name	Syntax
Directive	<%@ directive %>
Declaration	<%! declaration %>
Expression	<%= expression %>
Fragment codes	<% fragment codes %>
Comment	<%-- comment --%>

Fragments of code (scriptlet) are exactly what their name says parts of Java script code. JSP comments will not be visible to the user after generating sites.

The main drawback of this approach is that the application logic is "hardcoded" in the user interface, which means that when changing application logic, user interface should also be changed.

2.5. The combination of Java technology servlet and JavaServer Pages

In previous chapters it was shown how the use of servlet only is bad because the user interface is "hardcoded" in the servlet, and the use of JSP technology is bad because the application logic is "hardcoded" in the user interface. These problems are solved by combining these two technologies in order to split the Web application into layers. A typical division into three layers: the business layer, presentation layer and data layer. Business layer (business logic layer) is a layer responsible for the application logic, and is run through servlets. Presentation layer deals with the presentation of data and application state, using the JSP technology. It takes the results of the business layer using the map of attributes. Data layer is responsible for the persistence of data. It takes care of storing data in files or databases. Presentation layer must communicate with data layer exclusively via the business layer.

This architecture is very similar to the model, whose popularity in the development of applications began to grow in the seventies of the last century - the MVC architecture (Model - View - Controller) and MVC2. In the MVC architecture, the application is divided into business and presentation layer, which is further subdivided into *view* and *controller*. In MVC2 architecture, presentation layer is separated from the application logic, which is why the MVC2 architecture is similar to our three-layer model. Graphic display of MVC2 architecture is given in figure 1.

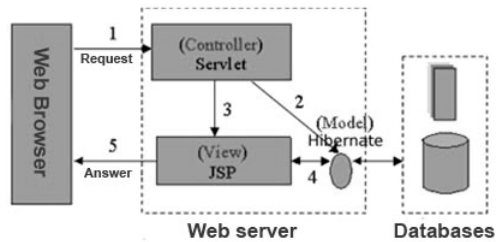

Figure 1 MVC2 architecture in the development of Web applications.

Code segment 3 An Example of application logic layer

```
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import
javax.servlet.http.HttpServletRequest;
import
javax.servlet.http.HttpServletResponse;
public class Controller extends HttpServlet
{@Override
protected void doGet
(HttpServletRequest req,
HttpServletResponse resp)
throws ServletException, IOException {
String sNumber =
(String)request.getParameter("number");
Integer no = null;
Integer result = null;
if(sNumber!=null) {
try {
no = Integer.valueOf(sNumber);
result = no.intValue()*intValue();
} catch(Exception ex) {}
}
else result = Integer.valueOf(0);
request.setAttribute("number", no);
request.setAttribute("result", result);
this.getServletContext().getRequest
Dispatcher("view.jsp").forward(req, resp);
}
}
```

Code segment 3 shows a simple servlet that takes care of application logic of the previous example with squaring

numbers. Servlet takes the parameter number, processes it, the result is sent to the attributes of requests map and further sends it to the file *view.jsp* on display by *getRequestDispatcher()* and *forward()* methods. The servlet does not even have a single HTML tag due to the fact that the *.jsp* database takes care of the user interface, while the servlet is used solely for calculating the results.

Code segment 4 an example of the presentation layer

```
<%@ page language="java"
session="false" contentType="text/html;
charset=UTF-8" %>
<html>
<head><title> Squares of
numbers</title></head>
<body><table cols="2" border="1">
<tr><td><b>Number</b></td><td><b>
Squares of numbers</b></td></tr>
<%
Integer number =
(Integer)request.getParameter("number");
Integer result =
(Integer)request.getParameter("result");
out.print("<tr><td>"+number+
"</td><td>"+result+"</td></tr>");
%>
    </table></body></html>
```

Code segment 4 shows the presentation layer in the form of a database *view.jsp* which takes over the servlet processing results of the previous examples. The result is downloaded over the attributes of requests map, as an attribute whose name is "*result*".

3. AN EXAMPLE OF A SIMPLE WEB APPLICATION IN THE THREE-LAYER ARCHITECTURE

The following example is a simple application made in the three-layer architecture. It is used for the input of the student enrollment data to database through a Web form. HTML language is used for creatiing the form. Through the form, user accesses the application, which represents user interface and in the three-layer architecture presentation layer. After the user enters data, servlet that processes business logic sets in. Servlet is a business layer in the three-layer architecture. It sends the data to the database, i.e. data layer, and generates a response which is then sent back to the user browser to be viewed.

3.1. The presentation layer

The presentation layer shows the Web form in which the user enters information about the student, and Figure 2 shows the layout of the form.

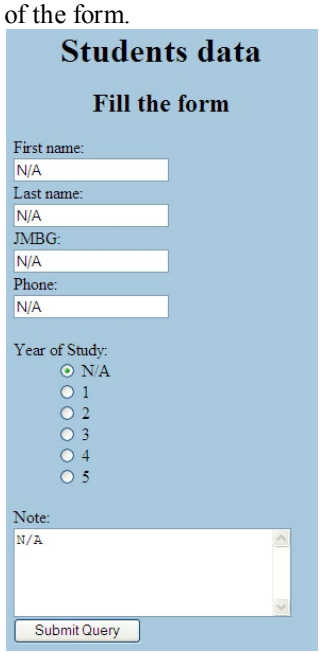

Figure 2 Students data form

It is possible in the appropriate fields to enter name, identification number, phone, year of study of the user and a note about the student. Predefined values of all elements are N / A, i.e. not available.

3.2. The business layer

After entering data into the form, the user presses the button that calls servlet. Servlet receives data entered by the user using the methods of

request.getParameter(). Then the data is stored in a database that is implemented in this application through class *Database*. Testing the data follows. If the user does not enter the identification number and last name, then the user is sent an appropriate message, and if the entry is correct, the data that the user has entered can be reviewed in the browser. Main part of the Servlet is the *doPost* method. It receives data and generates a response to the user. The *HttpServletRequest* class contains *doPost* method. Next to it, there is a similar method *doGet* which is used when receiving only one item, which is not the case with this application. The difference between them is that the method doGet can receive data from a static HTML document, while the *doPost* method can receive data from JSP documents only.

3.3. The data layer

The data layer in this case does not constitute of a database, but a class that is used for data entry.

Response that is needed to generate in the servlet can be achieved in several ways. Here it is made to call in the *PrintWriter* class and with the method println directly print HTML content into a Web browser. This is achieved with the following command:

```
response.setContentType("text/html");
```

Servlet makes it clear that the response is a HTML file. The answer is to be achieved by a servlet which calls in JSP file using the *RequestDispatcher* class. E.g.:

```
RequestDispatcher dispatcher =
getServletContext().getRequestDispatcher(
address);
dispatcher.forward(request, response);
```

The parameter address indicates the URL of the JSP file that we want to use. JSP file itself has the ability to access data from a servlet, but the aim of the three-layer architecture is the separation of the business layer from the presentation layer, therefore it is better that servlet sends the data received into the JSP database, which receives it and then processes. Data that we send to that database can also be transferred in several ways. The first way is by using the *HttpServletRequest* class which records the information we send to it by request.setAttribute("key1", value1); Where the key1 is name of the variable that the JSP database will receive by:

```
Type1 value1 = (Type1)
request.getAttribute("key1");
```

Data transfer is also possible using the JavaBean. [6] JavaBean is a class used for storing many classes so that they actually all make a class that can be transferred. More information about the JavaBean and other content can be found at http://java.sun.com/.

4. CONCLUSION

Development of a Web page today increasingly requires very good organization of data. Since the data increases, it is more and more difficult to create Web pages. Instead of ordinary static HTML pages, there are more dynamic Web pages that meet the demands of customers. The need to store large amounts of data and their further processing requires the construction of complex Web applications. Web applications must serve the needs of users, and be easy to use. Because of a number of different operating systems, there is a need for changing the application.

Development of Web applications in the Java programming language provides unlimited possibilities; this is why this technology is the right choice for many serious projects. Using three-layer architecture for creating Web pages greatly facilitates their maintenance because changes of one layer do not require changing the other layers, which makes the application easy to transfer.

REFERENCES:

[1] Java Servlet 3.0 Specification, http://jcp.org/aboutJava/communityprocess/pr/jsr315/index.html, 2009.
[2] JavaServer Pages 2.0 Specification, http://jcp.org/aboutJava/communityprocess/first/jsr152/index3.html, 2009.
[3] HTML 4.01 Specification, http://www.w3.org/TR/REC-html40/, 2009.
[4] Extensible Markup Language (XML) 1.0 (Fifth Edition), http://www.w3.org/TR/REC-xml/, 2009.
[5] Applet documentation from the Java Tutorial, http://java.sun.com/applets/, 2009.
[6] Zukowski, J.: Mastering Java 2 J2SE 1.4, Kompjuter Biblioteka, 2004.
[7] JavaServer Pages, http://en.wikipedia.org/wiki/JavaServer_Pages, 2009.
[8] Dr Siniša Vlajić: Napredni koncepti Jave i web programiranje u Javi, FON, Beograd 2005.
[9] M. Pirnau, Implementing Web Services Using Java Technology, International Journal of. Computers Communications & Control, vol.5, no.2.