Servlets and JDBC

1.Example of usage

1.1 Servlets

Servlets are Java technologies which open new perspectives on web servers. These facilities and advantages which make from Java an ideal programming language for writing client orientated applications, offer a good developing platform for server orientated applications[1].

Servlets are Java classes which run on WEB servers , between browser or HTTP client requests and data bases or applications from Hypertext Transport Protocol (HTTP) servers. Also, a servlet is a web component, managed by a container which generates dynamic content. Servlets interact with clients on request-response paradigm[2].

Although all servlet applications are made in Java, there are no restrictions for writing client component. Clients could be implemented in every programming language. Also, servlets could be clients in bigger network applications which imply complex services. For example, a servlet could use the facilities of JDBC for data base connectivity.

Life cycle of servlets is a specific property. Servlets are loading in a dynamic way. There is also the possibility to specify the load of servlets when the server starts. Once loaded, they become part of server. The process is transparent for users, he only has to specify the location of servlet, the class name which contains it and the servlet alias. After the servlets are loaded, there are three main stages : initialization, usage and destroy[3].

There are many possibilities of using the facilities offered by a servlet. A simple servlet can modify the information sent through a HTML format, should insert records in a database and it has the possibility to response the HTML request.

Servlets can generate dynamic HTML pages. Using *java.io.PrintWriter* this thing becomes very easy. There is no need of scripts to generate dynamic web pages. In Web server with Java support, servlets can be called for helping the pre-processing of pages. An example of calling servlets from browser is shown in *ex1*.

Servlets which are used together with HTTP protocol can implement and offer support for any of the following methods : GET, POST, HEAD etc. They are capable to redirect the requests to other locations or to generates specific errors for HTTP protocol. They can have access to all parameters includes by a HTML form. In second example , there is a servlet class which get parameters from a HTML form and shows the sum of *a*'s letters from First Name and Last Name (*ex2)*. Getting the parameters from HTML form is done by *getParameter* function.

Javax.servlet.http.HttpServlet class is at the base of hierarchy which implements HTTP servlets. It is responsible of solving caching problems and errors. In this example *doGet* contains two parameters : request and response. The first parameter contains data sent by client and the second allow servlets to report errors and to send a response to the client[4].

Servlets could be important components in multiserver applications. Not all the requests can be offered by a Web server. The web server is capable to receive the request and to send through a servlet, request to specialized servers. Another servlet will intercept the response for specialized server, to translate it in a specific format for the client which lunch the request. In *ex3* is shown a model of multiserver application, known by three – tier model.

First level is represented by browsers which send requests to a server (Java Web Server), the second level of the model[5].

1.2JDBC

JDBC (Java DataBase Connectivity) is a collection of classes which allow a client application wrote in Java to access a relational data bases server. Using these classes, an application can open a data base connectivity, create an object and send SQL queries through it and modify the results received from server. File sources which use JDBC classes should include java.sql[6].

To access DataBase server, for example MySQL, we have to make a connection. This connection is realised by an object of type Connection (interface which try to make a connection). The username and the password should be also send to validate the access[].

To execute SQL interrogations, it must be create an object with Statement interface. Then we create a String which is a valid SQL command, send the query and obtain the result of the interrogation, using one of the specific methods : *executeQuery(String str)*, where *str* is an instruction of type **SELECT** and returns an object of type *ResulSet* which contains the result of interrogation or *executeUpdate(string str)* where *str* is an instruction of type **INSERT, DELETE** or **UPDATE** which returns number of altered rows[7]**.**

In *ex4* is shown an example of a servlet which use JDBC API. The servlet gets the first name and last name from an HTML form and insert it in *BDD* data base (server MySQL), table *Persons*, if the record doesn't exist or return a message if the record already exists. The table contains an ID which is auto-increment primary key, a first_name field and a second_name field.

2.Advantages and disadvantages

2.1 Servlets

The **advantages** of Servlets are providing a way to generate dynamic HTML pages and the code is relative easy, providing all the powerfull features of Java programming language, such as exception handling and garbage collection. They also enable portability and can communicate with different servlet and servers.

The **disadvantages** of Servlets are designing (it is very difficult to make a good design for an application using servlets and it also slows down the process), complex business logic which can make the application unreadable , hard to understand. A programmer will also need a Java Runtime Environment on the server[8].

2.2 JDBC

The **advantages** of JDBC are performance, even if we are working with many records, easiness in processing SQL commands and syntax. JDBC technology is also easy to study and to implement. Abstractness brings by JDB ensure the independence of application from data base (Java code will be the same, only the driver will change). A programmer could

create XML structure of data from database automatically, no content conversion is required, can be use for Synchronous and also for Asynchronous processing and it supports modules.

The **disadvantage**s of JDBC technology are commands which are different from a database to another. Also, there is no aggregation and the MVC architecture is hard to implement. A programmer could not update or insert multiple tables with sequence and drivers have to be deployed for each type of database[9].

3. Examples of similar technologies

3.1 Servlets

CGI (Common Gateway Interface) defines a modality to change information with an extern application. Using CGI, Web server can access programs, interact with them and can make useful functions, such as data bases connectivity. CGI has also a few problems. Every time when Web server receive a request which needs CGI, server has to activate CGI (initializes it, executes it and returns the HTML results). User can integrate some libraries, but a bad wrote library could block the server[10].

PHP (Hypertext Preprocessor) is a programming language. Initially, it was used only for generate dynamic web pages, but now it is very useful for programmers to develop web pages and applications. PHP is a structural programming language and easy to understand, such as C, Perl or Java, the syntax of it is almost a combination between these three languages. It is also the most important programming web language open-source and server-side. There are versions available for the majority of web servers and for all operating system. In a regular PHP setup, every user request to a PHP application results in a separate instance. The request completes when the application completes. This means that PHP instances are short-lived, typically completing operations in a few seconds[11].

3.2 JDBC

ODBC is a collection of procedures wrote in C programming language and it is dependent of operating system .

JPA (Java Persistence API) is a specification of a programming interface of applications which describes the relational databases in programs that use Java Standard Edition (SE) and Java Enterprise Edition (JEE) platforms. It covers three domains : API (javax.persistence), Java Persistence Query Language (JPQL) and object/relational metadata.

Hibernate ORM is an object relational mapping framework. Hibernate's primary feature is mapping from Java classes to database tables (and from Java data types to SQL data types). Hibernate also provides data query and retrieval facilities. It generates SQL calls and relieves the developer from manual result set handling and object conversion. Applications using Hibernate are portable to supported SQL databases with little performance overhead[12]

Annexes

Ex1

```
<form method="GET" action ="Add">
<input type = "submit" value = "Add" class = "but">
</form>
```

Ex2

HTML :

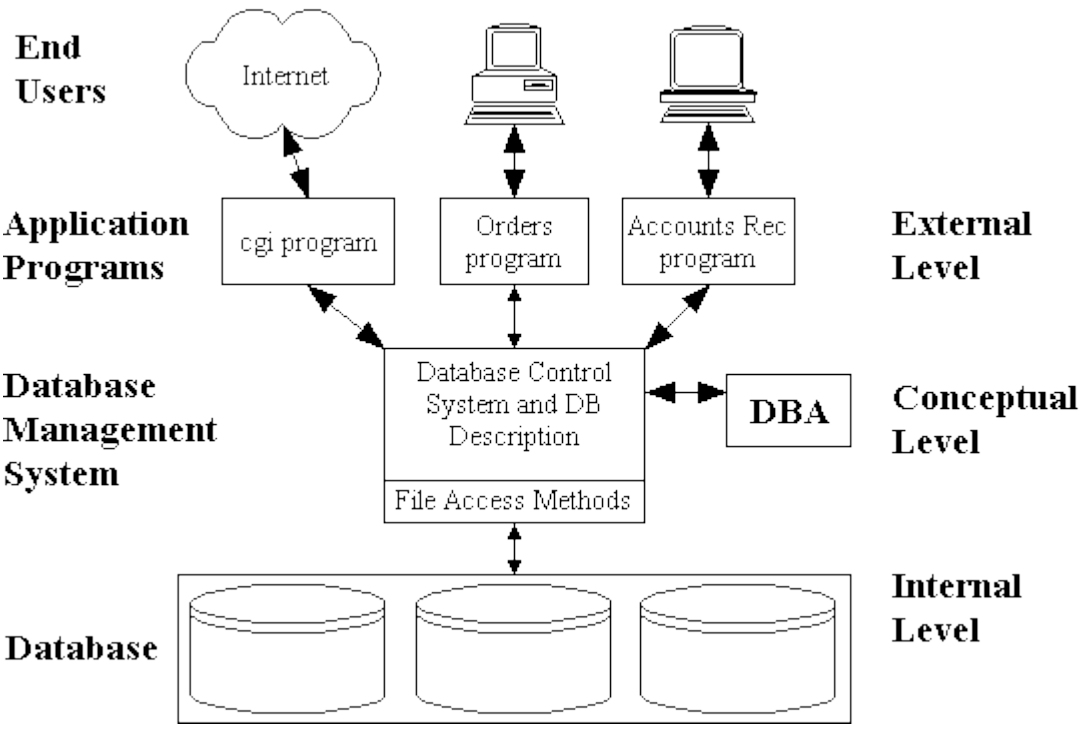
```
<form method="GET" action ="Add">
<input type = "text" value = "" name = “firstname” >  <br>
<input type  = “text” value = “” name = “lastname”> <br>
<input type = “submit”  name  = “Add”>
</form>
```

Servlet :

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class Test3 extends HttpServlet {

   protected void doGet(HttpServletRequest request, HttpServletRespo nse response)
         throws ServletException, IOException {

       response.setContentType("text/html;charset=UTF-8");
       PrintWriter out = response.getWriter();

      String fname = request.getParameter("firstname");
      String lname = request.getParameter("lastname");
      int number = 0;

      for(int i=0; i<fname.length(); i++)
```

```
        if(fname.charAt(i) == 'a')
            number++;
      for(int i=0; i<lname.length(); i++)
        if(lname.charAt(i) == 'a')
            number++;


        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Example 2</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h2> The sum of a's is " +number+ " </h2>");
        out.println("</body>");
        out.println("</html>");
   }
}
```

Ex3



Ex4

DataBase Class

```
import java.sql.*;

public class DataBase {

   private Connection conn;
   private  Statement state;
   private ResultSet results;

   public DataBase(){}

   public Connection Start(){

      try
                  {
            Class.forName("com.mysql.jdbc.Driver");
                  conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/BDD", "username",
"password");
            state = (Statement) conn.createStatement();
                  }
         catch(ClassNotFoundException cnfe)
         {
            System.out.println("* Driverul nu a putut fi incarcat! *");
            System.exit(1);
            return null;
         }
                  catch(SQLException sqlEx)
                  {
                        System.out.println("* Conectarea la baza de date a esuat! *");
                        System.exit(1);
               return null;
                  }

      return conn;
   }

   public void Stop(){
```

```
    try
                {
                        conn.close();
                }
                catch(SQLException sqlEx)
                {
                        System.out.println("* Nu m-am putut deconecta! *");
                        sqlEx.printStackTrace();
                }
  }

  public Statement getState(){

    return this.state;
  }

}
```

Query Class :

```
import java.sql.*;

public class Query extends BazaDate {

  ResultSet results;

  public Query(){

    Start();

  }

  public int add(String fname, String lname){

    int res = 0;

    try{
      Statement st = getState();
      String insert = "INSERT INTO Persons (first_name, last_name) VALUES
('"+fname+"', '"+lname+"')";
```

```
            res = st.executeUpdate(insert);
        }
        catch (SQLException e){
            System.out.println("Query couldn't be executed");
            e.printStackTrace();
            System.exit(1);
        }

        return res;
    }

    public ResultSet searcha(String fname, String lname){

        ResultSet res;

        try{
            Statement st = getState();
            String select = "SELECT * FROM Persons where  '"+fname+"' = first_name and
'"+lname+"' = last_name";
            res = st.executeQuery(select);
        }
         catch (SQLException e){
            System.out.println("Query couldn't be executed");
            e.printStackTrace();
            System.exit(1);
            res = null;
        }
        return res;
    }
}
```

Servlet

```
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;
import java.sql.ResultSet;
import java.util.logging.Level;
import java.util.logging.Logger;

public class Test4 extends HttpServlet {

   protected void doGet(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");

      String fname = request.getParameter("firstname");
      String lname = request.getParameter("lastname");
      int res = 0;
      ResultSet res1 = null;

      Query sql = new Query();


      try (PrintWriter out = response.getWriter()) {
         out.println("<!DOCTYPE html>");
         out.println("<html>");
         out.println("<head>");
         out.println("<title>Servlet Test4</title>");
         out.println("</head>");
         out.println("<body>");

         res1 = sql.search(fname, lname);
         if(!res1.next())
         {
            res = sql.adda(fname, lname);
            if(res == 0 )
               out.println("<h2> Could not insert </h2>");
            else out.println("<h2>Record inserted</h2>");
         }
         else{
            out.println("<h2> Record already exists  </h2>");
         }

         sql.close();
```

```
            out.println("</body>");
            out.println("</html>");
        } catch (SQLException ex) {
            Logger.getLogger(Test4.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

}
```

References

[1] “Dynamic HTML” A. Blaga

[2] “Thinking in Java” Prentince Hall

[3],[4] “Utilizare tehnici servlet” Marian Dobre

[5] “Programare - Servleturi: alternative la CGI” Clip P.

[6] Oracle JDBC Univestitatea Cluj

[7] JDBC – Horia Georgescu

[8] http://candidjava.com/

[9] http://www.tribunaeconomica.ro/

[10] “Utilizare tehnici servlet” Marian Dobre

[11] https://hibernate.atlassian.net/

[12] “Usage of server-side programming languages for websites”