



5.3. Source Code

This sentence describes the path we followed while writing our code the best:

“Write your program for people first, computers second”

In this section we do not give you all of the source code, because it weights some thousands lines. Here we just provide you with the code gist of JSIS.

5.3.1 Servlet/JSP Source code

```
/**
 * <p>Class Name: Login.java</p>
 * <p>Description: This class is a servlet and gets request from clients. Then
 * it creates a FilterManager to handle filtering and sends the request to
 * appropriate location.</p>
 * <p>Copyright: Copyright (c) 2003</p>
 * <p>Date : 15.01.2003
 * @author Nasif Ekiz
 * @version 1.0
 */

package servlets;

// Java Core Packages
import java.io.*;
import java.util.*;

// Java Extension Packages
import javax.servlet.*;
import javax.servlet.http.*;

import filtering.*;

public class Login extends HttpServlet {

    private static final String CONTENT_TYPE = "text/html";

    //Initialize global variables
    public void init() throws ServletException {
    }

    //Process the HTTP Post request
    public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        response.setContentType(CONTENT_TYPE);
        PrintWriter out = response.getWriter();
        FilterManager manager;
        RequestDispatcher dispatcher;
        // this section will be implemented later.....
    }
}
```

**FOR PUBLIC
RELEASE**

We did not put here code of all the classes and methods we used in JSIS for the reasons described in this document above.

This information is provided just to give you some information how JSIS was coded.

Feel free to test this code. Cannot be used or distributed without permission of JSIS developers.

*Upon request full source code can be e-mailed to those who requests (in your e-mail please describe us your reasons for this).



```
/*
 *try {
 * util.FormChecker formChecker = new util.FormChecker(request,response);
 * formChecker.checkForm();
 *}catch (util.FormException ex) {
 * ex.printStackTrace();
 *}
 */

// here we create a FilterManager and make filtering ...
manager = new FilterManager();
manager.addFilter(new LoginFilter());
manager.addFilter(new AgentFilter());

try {

    manager.applyFilters(request,response);
    // if filtering is successful
    dispatcher = request.getRequestDispatcher("controller?showpage=main");
    dispatcher.forward(request,response);

} catch (FilterException ex) {
    // if filtering is unsuccessful
    dispatcher = request.getRequestDispatcher("controller?showpage=error");
    dispatcher.forward(request,response);
}
} // end of doPost() method

public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

    doPost(request,response);

} // end of doGet() method

//Clean up resources
public void destroy() {
}
}

/**
 * <p>Class Name: FilterManager.java</p>
 * <p>Description: This class is used to manage the Filters and FilterChain. It
 * can create a new FilterChain and add some Filters to it.
 * <p>Copyright: Copyright (c) 2003</p>
 * <p>Date : 14.01.2003
 * @author Engin Tozal
 * @version 1.0
 */

package filtering;
```





```
// Java Extension Packages
import javax.servlet.*;
import javax.servlet.http.*;

import controller.*;

public class FilterManager {

    private FilterChain filterChain;

    // constructor
    public FilterManager() {
        filterChain = new FilterChain();
    }

    // this method is used to execute all Filters necessary filtering methods
    public void applyFilters(HttpServletRequest request,
                            HttpServletResponse response)
        throws ServletException, FilterException {

        filterChain.processFilters(request, response);
    } // end of applyFilters() method

    // this method is used to add a new Filter to the FilterChain
    public void addFilter(Filter f) {
        filterChain.addFilter(f);
    } // end of addFilter() method

    // this method is used to add a new Filter with a specified index to the
    // FilterChain
    public void addFilter(Filter f, int index) {
        filterChain.addFilter(f, index);
    } // end of addFilter() method
} // end of FilterManager class
```

**FOR PUBLIC
RELEASE**



5.3.2 J2ME Source code

```
/*
 * Class Name: JsisConnect.java
 * Description: Establishes connection with the Remote URL and retrieves its content
 * Copyright: Copyright (c) 2003
 *
 * @date : 06.02.2003
 * @author mm
 * @usedby Jsis.JsisMIDlet
 */

package Jsis;

import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
import javax.microedition.io.*;
import java.io.*;

public class JsisConnect {

    private String URL;
    private HttpURLConnection conn;
    private InputStream iStrm;
    private StringBuffer buffer;
    private String dataStr;
    private String cookie;

    /** JsisConnect Constructor called from MIDlet
     * It initializes all variables
     */
    public JsisConnect() {
        buffer = new StringBuffer();
        conn = null;
        iStrm = null;
        URL="";
        dataStr="";
        cookie="";
    }

    /**
     * Returns dataStr variable containing server response data
     */
    public String getData(){
        String dataStr_temp=dataStr;
        dataStr="";
        return dataStr_temp;
    }

    /**
     * Returns length of dataStr variable
     */
}
```





```
public long getDataLength(){
    return dataStr.length();
}

/*
 * Returns dataStr variable containing server response data
 */
public String getSessionID(){
    return cookie;
}

/*
 * Retrieves URL
 * Upon successful completion, Server Response of URL(usually Servlet)
 * is placed into dataStr variable and method returns true.
 * If there was an error placing the command, returns false;
 */
public boolean getURL(String url){
    URL=url;
    boolean resp;
    try {
        //returns true if successfully retrieved URL
        boolean completed = retrieveURL(url);

        if (completed) resp=true;
        else resp=false;

    } catch (Exception e) {
        System.out.println("JSisConnect: Exception " + e.toString());
        resp=false;
    }

    return resp;
}

/*
 * Prepare connection and streams then retrieve URL.
 * If URL is retrieved then returns true and content of URL
 * is put into dataStr. Otherwise, returns false
 */
@usedby getURL
Client Request
*/
private boolean retrieveURL(String url) throws IOException {

    try {

        conn = (HttpConnection) Connector.open(url);
        conn.setRequestMethod(HttpConnection.GET);
        conn.setRequestProperty("User-Agent",
            "Profile/MIDP-1.0 Configuration/CLDC-1.0");
```





```
// If there is a cookie already assigned from previous server request,
// use it this time also. Append it to the header going to Server
if (cookie != ""){
    System.out.println("JSisConnect: COOKIE FOUND!.." +
        "Appending to Server HEADER");

    conn.setRequestProperty("Cookie", cookie );
}

// Our first connection to the Server.
// If we don't have cookie, connect, then read cookie value
int rc = conn.getResponseCode();

if (cookie == ""){
    System.out.println("JSisConnect: NO COOKIE YET..." +
        "GETTING COOKIE FROM SERVER...");
    cookie = readCookie( conn );
}

// -----
// Get Server Response
// -----
iStrm = conn.openInputStream();

// If connection was accepted : conn.getResponseCode()
if (rc==HttpURLConnection.HTTP_OK) {

    System.out.println("-----");
    System.out.println("JSisConnect: Getting from Server...");

    int ch;
    while ((ch = iStrm.read()) != -1) {
        buffer.append((char) ch);
    }
    dataStr = buffer.toString();
    System.out.println("JSisConnect: Data from Server arrived...\n");
    return true;
}
else {
    // Error in Server Response
    return false;
}
} finally {
    if(iStrm!= null) {
        // close Data Stream
        iStrm.close();
    }
    if(conn != null) {
        // close Connection
        conn.close();
    }
}
```





```
    }
}

/* Iterates through the
 * response headers and looks for a Set-Cookie
 * header. It then splits the header value
 * into parts and looks for a cookie value
 * with the name "JSESSIONID", which is what
 * a J2EE server will use for session tracking.
 */
private String readCookie( HttpURLConnection conn ) throws IOException {
    String key;
    String value;
    String[] substrs;

    for (int i = 0; ( key = conn.getHeaderFieldKey(i) ) != null; ++i){

        key = key.toLowerCase();
        System.out.println("HEADER "+i+" key: "+key+ " value: "+conn.getHeaderField( i
));

        if (key.equals("set-cookie")) {
            value = conn.getHeaderField(i);

            while (value != null) {
                substrs = split(value, ';');
                if(substrs[0].startsWith("JSESSIONID=")){
                    return substrs[0];
                }
                value = substrs[1];
            }
        }
    }
    return null;
}

/*
 * Splits given input String using value indicated by parameter "ch"
 */
private static String[] split(String in, char ch){
    String[] result = new String[2];
    int pos = in.indexOf(ch);
    if(pos != -1){
        result[0] = in.substring(0, pos).trim();
        result[1] = in.substring(pos+1).trim();
    } else {
        result[0] = in.trim();
    }
    return result;
}
}
```

