Ftpcon.java

package algorithm;

import java.io.File;

import java.io.FileInputStream;

import org.apache.commons.net.ftp.FTPClient;

public class Ftpcon {

FTPClient client = new FTPClient();

FileInputStream fis = null;

boolean status;

public boolean upload(File file){

try{

// client.enterLocalPassiveMode();

client.connect("ftp.drivehq.com");

client.login("igeeks", "igeeks123");

client.enterLocalPassiveMode();

//String filename = "/home/ibn/Desktop/report.txt";

fis = new FileInputStream(file);

status= client.storeFile(" /nandu/"+file.getName(), fis);

client.logout();

fis.close();

}

catch(Exception e){

System.out.println(e);

}

if(status){

System.out.println("success");

return true;

}

else{

System.out.println("failed");

return false;

}

}

}

package algorithm;

import java.io.InputStream;

import java.security.spec.KeySpec;

import javax.crypto.Cipher;

import javax.crypto.SecretKey;

import javax.crypto.SecretKeyFactory;

import javax.crypto.spec.DESedeKeySpec;

import org.apache.tomcat.util.codec.binary.Base64;

public class RS\_IBE {

private static final String UNICODE\_FORMAT = "UTF8";

public static final String DESEDE\_ENCRYPTION\_SCHEME = "DESede";

private KeySpec ks;

private SecretKeyFactory skf;

private Cipher cipher;

byte[] arrayBytes;

private String myEncryptionKey;

private String myEncryptionScheme;

SecretKey key;

public RS\_IBE() throws Exception {

myEncryptionKey = "ThisIsSpartaThisIsSparta";

myEncryptionScheme = DESEDE\_ENCRYPTION\_SCHEME;

arrayBytes = myEncryptionKey.getBytes(UNICODE\_FORMAT);

ks = new DESedeKeySpec(arrayBytes);

skf = SecretKeyFactory.getInstance(myEncryptionScheme);

cipher = Cipher.getInstance(myEncryptionScheme);

key = skf.generateSecret(ks);

}

public String encrypt(String unencryptedString) {

String encryptedString = null;

try {

cipher.init(Cipher.ENCRYPT\_MODE, key);

byte[] plainText = unencryptedString.getBytes(UNICODE\_FORMAT);

byte[] encryptedText = cipher.doFinal(plainText);

encryptedString = new String(Base64.encodeBase64(encryptedText));

} catch (Exception e) {

e.printStackTrace();

}

return encryptedString;

}

public String decrypt(String encryptedString) {

String decryptedText=null;

try {

cipher.init(Cipher.DECRYPT\_MODE, key);

byte[] encryptedText1 = encryptedString.getBytes(UNICODE\_FORMAT);

byte[] encryptedText = Base64.decodeBase64(encryptedText1);

byte[] plainText = cipher.doFinal(encryptedText);

decryptedText= new String(plainText);

} catch (Exception e) {

e.printStackTrace();

}

return decryptedText;

}

}

Upload.java

package network;

import com.oreilly.servlet.MultipartRequest;

import com.sun.org.apache.xerces.internal.impl.dv.util.Base64;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.Statement;

import java.text.DateFormat;

import java.text.SimpleDateFormat;

import java.util.Calendar;

import java.util.Date;

import javax.crypto.KeyGenerator;

import javax.crypto.SecretKey;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import Dbcon.DbConnection;

import algorithm.Ftpcon;

import algorithm.RS\_IBE;

import java.security.SecureRandom;

import java.util.Random;

public class Upload\_new extends HttpServlet {

File file;

final String filepath="D:/";

protected void processRequest(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

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

PrintWriter out = response.getWriter();

try {

HttpSession session = request.getSession();

String provider= (String) session.getAttribute("sssname");

MultipartRequest m=new MultipartRequest(request,filepath);

File file=m.getFile("file");

String filename=file.getName().toLowerCase();

System.out.println("File name"+filename);

Connection con= DbConnection.getConnection();

Statement st3=con.createStatement();

ResultSet rt3=st3.executeQuery("select \* from upload where filename='"+filename+"'");

if(rt3.next()){

response.sendRedirect("file\_upload.jsp?failed='yes'");

}

else{

//out.println("file location:"+filepath+"\n file name:"+filename+"\n");

BufferedReader br=new BufferedReader(new FileReader(filepath+filename));

StringBuffer sb=new StringBuffer();

String temp=null;

while(( temp=br.readLine())!=null){

sb.append(temp);

}

System.out.println("string Buffer"+sb);

RS\_IBE e=new RS\_IBE();

String IBE=e.encrypt(sb.toString());

//storing encrypted file

FileWriter fw=new FileWriter(file);

fw.write(IBE);

fw.close();

System.out.println("string Buffer"+IBE);

//Secret Key

Random RANDOM = new SecureRandom();

int PASSWORD\_LENGTH = 10;

String letters = "1234567890qwertyuioplkjhgfdsazxcvbnm1234567890";

String skey = "";

for (int i=0; i<PASSWORD\_LENGTH; i++)

{

int index = (int)(RANDOM.nextDouble()\*letters.length());

skey += letters.substring(index, index+1);

}

// date and Time

DateFormat dateFormat = new SimpleDateFormat("yyyy.MM.dd G 'at' HH:mm:ss ");

Date date = new Date();

String time= dateFormat.format(date);

System.out.println("current Date "+time);

String len=file.length()+"bytes";

//uploading file

boolean status=new Ftpcon().upload(file);

if(status){

//Connection con= Dbconnection.getConn();

Statement st=con.createStatement();

int i=st.executeUpdate("insert into upload(filename,content,provider\_name,time,secret\_key,status)values('"+file.getName()+"','"+IBE+"','"+provider+"','"+time+"','"+skey+"','No')");

System.out.println(i);

if(i!=0){

// out.println("success");

response.sendRedirect("file\_upload.jsp?status='uploded'");

}

else{

out.println("error while uploading additional informations");

}

// out.println("file stored");

// out.println(file.length());

}

else{

out.println("error");

}

}

}

catch(Exception e){

out.println(e);

}

finally {

out.close();

}

}

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">

/\*\*

\* Handles the HTTP

\* <code>GET</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Handles the HTTP

\* <code>POST</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Returns a short description of the servlet.

\*

\* @return a String containing servlet description

\*/

@Override

public String getServletInfo() {

return "Short description";

}// </editor-fold>

}