**COMP. 7214**

**Client Server Web Development**

**DESIGN ASSIGNMENT**

**Assignment 2**

**SUBMITTED TO: Doug Mackenzie**

**SUBMITTED BY: Akvinder Kaur**

Contents

[1. Full Stack Development 3](#_Toc51923859)

[2. REST API 5](#_Toc51923860)

[3. Documents Databases 18](#_Toc51923861)

[4. Single Page application 20](#_Toc51923862)

[5. Design Patterns 21](#_Toc51923863)

**DESIGN A SINGLE PAGE WEB APPLICATION**

# Full Stack Development

Back-end Development alludes to worker side turn of events. It is the term utilized for the in the background exercises that happen when playing out any activity on a site. As such, it is the part of programming that doesn't come in direct contact with the clients. Clients by implication access the parts created by Back-end creators through a Front-end application. Exercises, such as composing APIs, making libraries and working with framework parts without UIs are a few instances of Back-end advancement.

Full stack advancement is a piece of web improvement which comprehensively alludes to the errands related with creating sites for facilitating by means of intranet or web. It is the improvement of a total application both the front end, which we allude to as the customer side, and back-end the worker side. Because of the presentation of distributed computing there have been radical changes in the field of Full Stack Development and how it has influenced the turn of events. Distributed computing comprises of different administrations and has numerous favorable circumstances.

This kind of model depicts the capacities that must be finished at a specific layer however doesn't indicate accurately how a capacity ought to be practiced. A reference model isn't expected to give an adequate degree of detail to characterize absolutely how every convention should function at each layer. The basic role of a reference model is to help in a more clear comprehension of the capacities and cycles vital for network correspondences.

A web stack, likewise called a web application stack, is an accumulation of a product arrangement particularly for executing sites and web applications. The assortment of layers is known as a stack. The term, 'stack', alludes to the way that the framework's segments are based upon each other. The essential prerequisites important to build a web stack incorporate a working framework, a web worker, an information base, and a content mediator. Along with the correct worker equipment, this heap of parts guarantees that essential data about comparable web ventures is sent to mentioning customers which is commonly the web program.

Organizations typically went for full-stack improvement, however the time taken by the masters would be lesser than that of a group comprising of just full-stack engineers. To back this announcement up let us believe the profitability to be 1 unit for one case of time for a Full Stack Developer presently consider the all out efficiency required is four units, at that point it would require four occasions of time. While for an expert, let us consider two units of profitability for each example at that point to arrive at four units, it would take a less measure of time and subsequently is more productive. On the off chance that any inconsistencies happen while the association of the Back-end to the front you could include another example which would at present be quicker.

Distributed computing tackles noteworthy issues in business. Getting numerous advantages through distributed computing is turning into the new typical. [10] 1. You pay what you use: While utilizing distributed computing administrations, you just compensation for the administrations you use, which makes it more cost-effective 2. Adaptability: As your business advances the size of the information to be taken care of increment or diminishes cloud administrations give to change your cloud limit at whenever. 3. Reinforcements/information recuperation: For little scope organizations it is hard to set up an information recuperation frameworks yet distributed computing gives information recuperation choice which keeps away from a significant venture. 4. Portable Connect: One of the fundamental highlights of distributed computing is you can work from anyplace. 5. Security: When it comes to information, security consistently goes under thought as each client needs their information to be secure. The cloud specialist co-ops give the security of information, and the information in the cloud is safer contrasted with putting away the information in hard drives since loss of touchy information for little scope business having lacking assets to make sure about information from programmers and penetrates would be heartbreaking. 6. Site the board: Cloud figuring highlights incorporate information the executives and investigation. It enables you to oversee DNS workers, web administrations and so forth.

# REST API

Get the id of customer’s registration and the information of their registration.

**REST API: GET/customers**

import com.messagebird.MessageBirdClient;

import com.messagebird.MessageBirdService;

import com.messagebird.MessageBirdServiceImpl;

import com.messagebird.exceptions.GeneralException;

import com.messagebird.exceptions.UnauthorizedException;

import com.messagebird.exceptions.NotFoundException;

public class ExampleCancelNumber {

public static void main(String[] args) {

if (args.length < 2) {

System.out.println("Please specify your access key & the number you wish to delete.");

return;

}

// First create your service object

final MessageBirdService wsr = new MessageBirdServiceImpl(args[0]);

// Add the service to the client

final MessageBirdClient messageBirdClient = new MessageBirdClient(wsr);

try {

messageBirdClient.cancelNumber(args[1]);

System.out.println("Number Deleted!");

} catch (UnauthorizedException | GeneralException | NotFoundException exception) {

if (exception.getErrors() != null) {

System.out.println(exception.getErrors().toString());

}

exception.printStackTrace();

}

}

}

**REST API: POST/customers**

import com.messagebird.MessageBirdClient;

import com.messagebird.MessageBirdService;

import com.messagebird.MessageBirdServiceImpl;

import com.messagebird.exceptions.GeneralException;

import com.messagebird.exceptions.UnauthorizedException;

import com.messagebird.objects.conversations.ConversationContent;

import com.messagebird.objects.conversations.ConversationContentType;

import com.messagebird.objects.conversations.ConversationFallbackOption;

import com.messagebird.objects.conversations.ConversationSendRequest;

import com.messagebird.objects.conversations.ConversationSendResponse;

public class ExampleConversationSendMessage {

public static void main(String[] args) {

if (args.length < 3) {

System.out.println("Please at least specify your access key, the channel id and destination address.\n" +

"Usage : java -jar <this jar file> test\_accesskey(Required) channel\_id(Required) to(Required) fallback\_channel\_id(optional)");

return;

}

//First create your service object

final MessageBirdService wsr = new MessageBirdServiceImpl(args[0]);

//Add the service to the client

final MessageBirdClient messageBirdClient = new MessageBirdClient(wsr);

ConversationFallbackOption fallbackOption = null;

if (args.length == 4) {

fallbackOption = new ConversationFallbackOption(args[3], "5m");

}

ConversationContent conversationContent = new ConversationContent();

conversationContent.setText("Hello world from java sdk");

ConversationSendRequest request = new ConversationSendRequest(

args[1],

ConversationContentType.TEXT,

conversationContent,

args[2],

"",

fallbackOption);

try {

ConversationSendResponse sendResponse = messageBirdClient.sendMessage(request);

System.out.println(sendResponse.toString());

} catch (GeneralException | UnauthorizedException exception) {

exception.printStackTrace();

}

}

}

Login to an account

Every customer must follow this process to access their own account safely and not to login the account of someone’s else.

**REST API: POST/customers**

import com.messagebird.MessageBirdClient;

import com.messagebird.MessageBirdService;

import com.messagebird.MessageBirdServiceImpl;

import com.messagebird.exceptions.GeneralException;

import com.messagebird.exceptions.UnauthorizedException;

import com.messagebird.objects.voicecalls.VoiceCallFlowRequest;

import com.messagebird.objects.VoiceStep;

import java.util.Collections;

public class ExampleCreateVoiceCallFlow {

public static void main(String[] args) {

if (args.length < 2) {

System.out.println("Please specify your access key and voice call flow arguments");

return;

}

//First create your service object

final MessageBirdService wsr = new MessageBirdServiceImpl(args[0]);

//Add the service to the client

final MessageBirdClient messageBirdClient = new MessageBirdClient(wsr);

final VoiceCallFlowRequest voiceCallFlowRequest = new VoiceCallFlowRequest();

voiceCallFlowRequest.setTitle(args[1]);

voiceCallFlowRequest.setRecord(true); // Can be false as well, see docs

VoiceStep voiceStep = new VoiceStep();

voiceCallFlowRequest.setSteps(Collections.singletonList(voiceStep)); // VoiceStep Object

voiceCallFlowRequest.setDefaultCall(true); // Can be false as well, see docs

try {

//Creating voice call by id

System.out.println("Creting a Voice Call Flow");

messageBirdClient.sendVoiceCallFlow(voiceCallFlowRequest);

System.out.println("Voice call flow created");

} catch (GeneralException | UnauthorizedException exception) {

exception.printStackTrace();

}

}

}

If customer creates their profile after the registration the customer should have its own basic detail. Still the little difference is the profiles will build up the arrays and fill the detail based on what the customer has provided while registration.

**REST API: POST/profiles**

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.SQLException;

import java.text.SimpleDateFormat;

import java.util.Date;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

\* Servlet implementation class CreateAssistant

\*/

@WebServlet("/CreateAssistant")

public class CreateAssistant extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

public CreateAssistant() {

super();

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

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

response.getWriter().append("Served at: ").append(request.getContextPath());

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

doGet(request, response);

String name = request.getParameter("name");

String email = request.getParameter("email");

String phone = request.getParameter("phone");

String pwd = request.getParameter("pwd");

String joindate = new SimpleDateFormat("yyyy-MM-dd").format(new Date());

PrintWriter out = response.getWriter();

Connection c;

try {

c = DriverManager.getConnection("jdbc:mysql://localhost:3306/hospital","root","root");

String sql = "insert into assistant(name,email,phone,joindate,password) values(?,?,?,?,?)";

PreparedStatement ps = c.prepareStatement(sql);

ps.setString(1,name);

ps.setString(2,email);

ps.setString(3,phone);

ps.setString(4,joindate);

ps.setString(5,pwd);

ps.addBatch();

// Executing SQL

int successCount = 0;

successCount += ps.executeBatch()[0];

ps.clearBatch();

if(successCount == 1) {

response.sendRedirect("login.html");

}

else {

response.setContentType("text/html");

out.println("<br><br><br><h1 align=center><font color=\"red\">TRY AGAIN<br>REDIRECTING BACK REGISTERATION PAGE</font></h1><script type=\"text/javascript\">");

out.println("redirectURL = \"newAssistant.html\";setTimeout(\"location.href = redirectURL;\",\"5000\");");

out.println("</script>");

}

} catch (SQLException e) {

e.printStackTrace();

response.setContentType("text/html");

out.println("<br><br><br><h1 align=center><font color=\"red\">TRY AGAIN<br>REDIRECTING BACK REGISTERATION PAGE</font></h1><script type=\"text/javascript\">");

out.println("redirectURL = \"newAssistant.html\";setTimeout(\"location.href = redirectURL;\",\"5000\");");

out.println("</script>");

}

}

}

If the user wants to create an advertisement or a new advertisement in the profiles array and I use PUT method because with the help of PUT method, we can have more than one advertisement unlike the profile, and it can have only one array.

**REST API: PUT/contents/1**

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.ResultSetMetaData;

import java.sql.SQLException;

import java.sql.Statement;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

\* Servlet implementation class RetrievePatientsDID

\*/

@WebServlet("/RetrievePatientsDID")

public class RetrievePatientsDID extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

public RetrievePatientsDID() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

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

// response.getWriter().append("Served at: ").append(request.getContextPath());

PrintWriter out = response.getWriter();

Connection c = GetConnection.getConnection();

String did = request.getParameter("did");

System.out.println(did);

String sql = "select patients,name from doctor where did = "+did;

try {

Statement s = c.createStatement();

ResultSet r = s.executeQuery(sql);

r.next();

String[] pList = r.getString("patients").split(",");

String name = r.getString("name");

ResultSet rrr = s.executeQuery("select \* from patient");

ResultSetMetaData rms = rrr.getMetaData();

response.setContentType("text/html");

out.println(" <style> table {font-family: arial, sans-serif;border-collapse: collapse;width: 100%;} td, th {border: 1px solid #dddddd;text-align: left;padding: 8px;}tr:nth-child(even) {background-color: #dddddd;} </style>");

out.println("<h2>List of all the Patients working under: "+name+"</h2>");

out.println("<table>");

out.println("<tr>");

out.println("<th>"+rms.getColumnName(1)+"</th>");

out.println("<th>"+rms.getColumnName(2)+"</th>");

out.println("<th>"+rms.getColumnName(3)+"</th>");

out.println("<th>"+rms.getColumnName(4)+"</th>");

out.println("<th>"+rms.getColumnName(5)+"</th>");

out.println("<th>"+rms.getColumnName(6)+"</th>");

out.println("<th>"+rms.getColumnName(7)+"</th>");

out.println("<th>"+rms.getColumnName(8)+"</th>");

out.println("<th>"+rms.getColumnName(9)+"</th>");

out.println("<th>"+rms.getColumnName(10)+"</th>");

out.println("<th>"+rms.getColumnName(11)+"</th>");

out.println("</tr>");

for(String p: pList) {

if(Integer.valueOf(p) >= 0){

Statement ss = c.createStatement();

ResultSet rr = ss.executeQuery("select \* from patient where pid = "+p);

rr.next();

out.println("<tr>");

out.println("<td>"+rr.getString(1)+"</td>");out.println("<td>"+rr.getString(2)+"</td>");out.println("<td>"+rr.getString(3)+"</td>");out.println("<td>"+rr.getString(4)+"</td>");

out.println("<td>"+rr.getString(5)+"</td>");out.println("<td>"+rr.getString(6)+"</td>");out.println("<td>"+rr.getString(7)+"</td>");out.println("<td>"+rr.getString(8)+"</td>");

out.println("<td>"+rr.getString(9)+"</td>");out.println("<td>"+rr.getString(10)+"</td>");out.println("<td>"+rr.getString(11)+"</td>");

out.println("</tr>");

}

}

out.println("</table>");

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

doGet(request, response);

}

}

User should know the id of a customer but not the login id before getting their content of advertisement. When the user wants to get their content of advertisement then, every customer has their own content of their advertisement in their profile. Afterwards, Data structure of JSON should have the information of the customer with a profile array and it contains the information of customer’s profile inside the profile array which includes the advertisement and the advertisement will have its own array.

**REST API: GET/profiles/contents/1**

import com.messagebird.MessageBirdClient;

import com.messagebird.MessageBirdService;

import com.messagebird.MessageBirdServiceImpl;

import com.messagebird.exceptions.GeneralException;

import com.messagebird.exceptions.NotFoundException;

import com.messagebird.exceptions.UnauthorizedException;

/\*\*

\* Created by rvt on 1/8/15.

\*/

public class ExampleDeleteMessage {

public static void main(String[] args) {

if (args.length == 0) {

System.out.println("Please specify your access key example and a id to delete: java -jar <this jar file> test\_accesskey 0f15f050454ad3db04286d6b30005106");

return;

}

// First create your service object

final MessageBirdService wsr = new MessageBirdServiceImpl(args[0]);

// Add the service to the client

final MessageBirdClient messageBirdClient = new MessageBirdClient(wsr);

try {

// Deleting message by id

System.out.println("Delete message:");

messageBirdClient.deleteMessage(args[1]);

System.out.println("Message ["+args[1]+"] deleted.");

} catch (UnauthorizedException | GeneralException | NotFoundException exception) {

if (exception.getErrors() != null) {

System.out.println(exception.getErrors().toString());

}

exception.printStackTrace();

}

}

}

**REST API: GET/profiles/contents/1**

import com.messagebird.MessageBirdClient;

import com.messagebird.MessageBirdService;

import com.messagebird.MessageBirdServiceImpl;

import com.messagebird.exceptions.GeneralException;

import com.messagebird.exceptions.NotFoundException;

import com.messagebird.exceptions.UnauthorizedException;

public class ExampleDeleteRecording {

public static void main(String[] args) {

if (args.length < 3) {

System.out.println("Please specify your access key and a call\_id, leg\_id, and recording\_id to delete: java -jar <this jar file> <test\_accesskey> <call\_id> <leg\_id> <recording\_id>" );

return;

}

// First create your service object

final MessageBirdService wsr = new MessageBirdServiceImpl(args[0]);

// Add the service to the client

final MessageBirdClient messageBirdClient = new MessageBirdClient(wsr);

try {

// Deleting message by id

System.out.println("Delete recording:");

messageBirdClient.deleteRecording(args[1],args[2],args[3]);

System.out.println("Recording ID ["+args[3]+"] deleted.");

} catch (UnauthorizedException | GeneralException | NotFoundException exception) {

if (exception.getErrors() != null) {

System.out.println(exception.getErrors().toString());

}

exception.printStackTrace();

}

}

}

**REST API: DELETE/customers/1**

{ import com.messagebird.MessageBirdClient;

import com.messagebird.MessageBirdService;

import com.messagebird.MessageBirdServiceImpl;

import com.messagebird.exceptions.GeneralException;

import com.messagebird.exceptions.UnauthorizedException;

import com.messagebird.exceptions.NotFoundException;

public class ExampleDeleteVoiceCall {

public static void main(String[] args) {

if (args.length < 2) {

System.out.println("Please specify your access key and a call ID : java -jar <this jar file> test\_accesskey e8077d803532c0b5937c639b60216938");

return;

}

//First create your service object

final MessageBirdService wsr = new MessageBirdServiceImpl(args[0]);

//Add the service to the client

final MessageBirdClient messageBirdClient = new MessageBirdClient(wsr);

try {

//Deleting voice call by id

System.out.println("Deleting voice call");

messageBirdClient.deleteVoiceCall(args[1]);

System.out.println("Voice call [" + args[1] + "] deleted.");

} catch (GeneralException | NotFoundException | UnauthorizedException exception) {

exception.printStackTrace();

}

}

}

|  |
| --- |
|  |
|  |
|  |

# Documents Databases

\*\*

\* encoding: UTF-8

\* JaC64 - Application for JaC64 emulator

\* A Swing UI for the JaC64 emulator for download to Java enabled

\* Desktop computers.

\* Created: Sat Dec 08 23:27:15 2007

\*

\* @author Joakim Eriksson, Dreamfabric / joakime@sics.se

\* @version 1.0

\*/

import java.applet.\*;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

import javax.swing.event.TableModelEvent;

import javax.swing.table.AbstractTableModel;

import javax.swing.table.TableModel;

import java.net.\*;

import com.dreamfabric.jac64.\*;

import com.dreamfabric.c64utils.\*;

public class JaC64 implements ActionListener, KeyEventDispatcher {

private static final String ABOUT\_MESSAGE =

"JaC64 version: " + C64Screen.version + "\n" +

"JaC64 is a Java-based C64 emulator by Joakim Eriksson\n" +

"The SID emulation use the resid Java port by Ken Händel\n\n" +

"For more information see: http://www.jac64.com/";

private C64Reader reader;

private C64Screen scr;

private boolean fullscreen = false;

private CPU cpu;

private JFrame C64Win;

private KeyListener c64Canvas;

private FileDialog fileDialog;

private JMenuItem load;

private JTable fileTable;

private JDialog loadFile;

private DirEntry[] dirEntries;

private static final String[] SID\_TYPES = new String[] {"SID: resid MOS 6581",

"SID: resid MOS 8580", "SID: JaC64 Original"};

private static final String[] JOYSTICK = new String[] {"Joystick in port 1",

"Joystick in port 2"};

private TableModel dataModel = new AbstractTableModel() {

public final String[] NAMES = new String[] {"File name", "Size", "Type"};

public int getColumnCount() { return 3;}

public int getRowCount() { return (dirEntries != null ? dirEntries.length : 0);}

public Object getValueAt(int row, int col) {

if (col == 0) return dirEntries[row].name;

if (col == 1) return new Integer(dirEntries[row].size);

if (col == 2) return dirEntries[row].getTypeString();

return "-";

}

public String getColumnName(int col) {

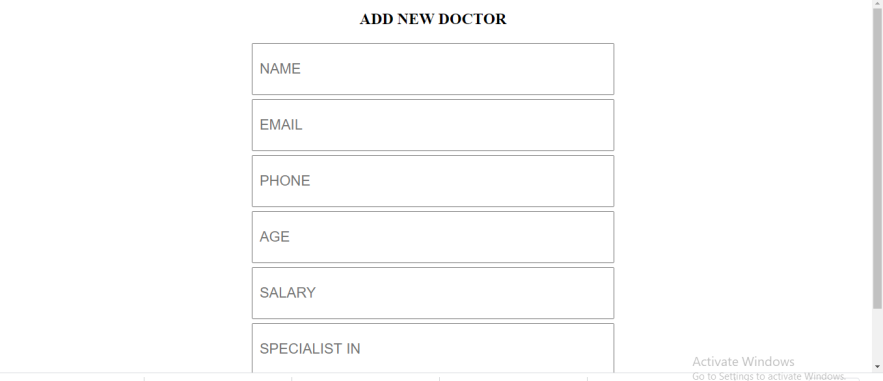
return NAMES[col];

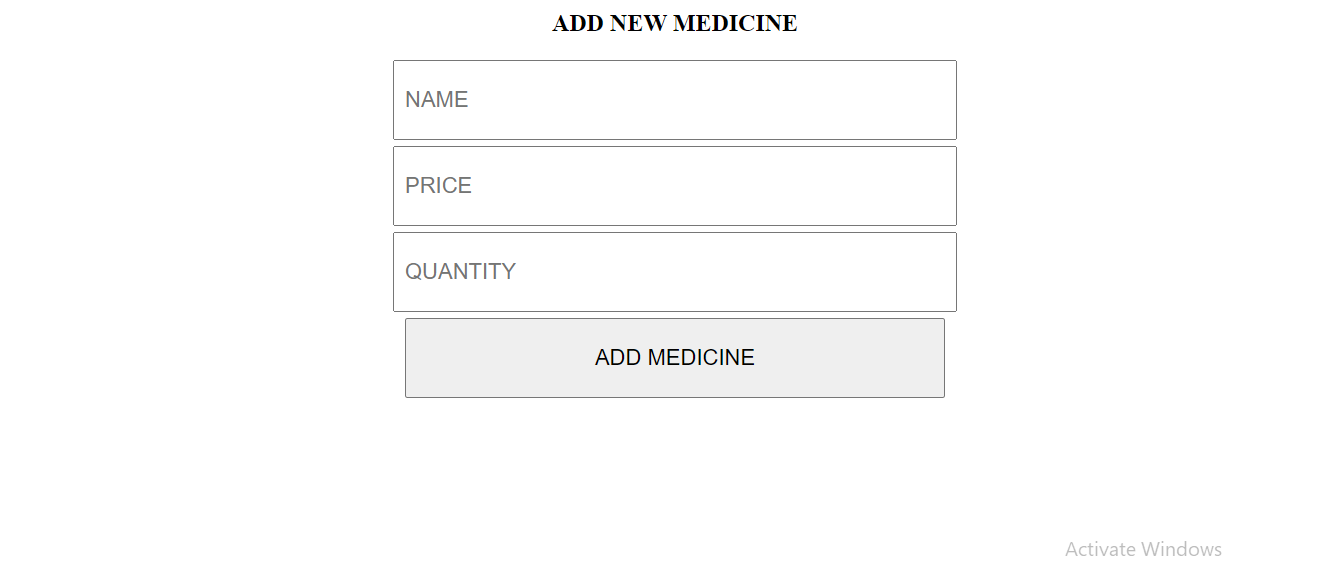
}

};

# Single Page application

|  |  |
| --- | --- |
| **Template Name** **:** | Free single Page Medical Hospital Website Template |
| **License** **:** | Free License Under Creative Commons Attribution 3.0  Unlimited Use, you can help & support  Liontemplates by donations or you should keep link to our website. |
| **Date Created :** | 02 -01 – 2019 |
| **Compatible Browsers:** | Google Chrome, Firefox, Safari, IE 10, Opera etc. |
| **Source Files included:** | HTML, Java, Bootstrap 4, JS, CSS, Fonts, Jquery,SCSS |
| **Mobile Responsive :** | Yes. |





# Design Patterns

public JaC64() {

SIDMixer.DL\_BUFFER\_SIZE = 16384;

Debugger monitor = new Debugger();

cpu = new CPU(monitor, "", new SELoader());

scr = new C64Screen(monitor, true);

cpu.init(scr);

// Reader available after init!

scr.init(cpu);

scr.registerHotKey(KeyEvent.VK\_BACK\_SPACE, KeyEvent.CTRL\_DOWN\_MASK |

KeyEvent.ALT\_DOWN\_MASK

, "reset()", cpu);

scr.registerHotKey(KeyEvent.VK\_F12, KeyEvent.CTRL\_DOWN\_MASK

, "toggleFullScreen()", this);

reader = new C64Reader(); // scr.getDiskDrive().getReader();

reader.setCPU(cpu);

cpu.getDrive().setReader(reader);

C64Win = new JFrame("JaC64 - A Java C64 Emulator");

C64Win.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JMenuBar jbar = new JMenuBar();

C64Win.setJMenuBar(jbar);

JMenu filem;

JMenuItem mi;

jbar.add(filem = new JMenu("File"));

filem.add(mi = new JMenuItem("Open File/Disk"));

mi.addActionListener(this);

filem.add(load = new JMenuItem("Load File"));

load.addActionListener(this);

filem.add(mi = new JMenuItem("Reset"));

mi.addActionListener(this);

filem.add(mi = new JMenuItem("Hard Reset"));

mi.addActionListener(this);

filem.add(mi = new JMenuItem("About JaC64"));

mi.addActionListener(this);

jbar.add(filem = new JMenu("Settings"));

JMenu subm;

filem.add(subm = new JMenu("Color Set"));

createRadioMenu(subm, new String[] {"Color Set 1 - JaC64 original",

"Color Set 2 - darker", "Color Set 3 - softer",

"Color Set 4 - Win VICE"}, 0);

filem.add(subm = new JMenu("SID Emulation"));

createRadioMenu(subm, SID\_TYPES, 0);

filem.add(subm = new JMenu("Joystick Port"));

createRadioMenu(subm, JOYSTICK, 0);

C64Win.setBackground(Color.black);

C64Win.setForeground(Color.black);

C64Win.setLayout(new BorderLayout());

C64Win.getContentPane().add(scr.getScreen(),

BorderLayout.CENTER);

C64Win.setFocusable(true);

C64Win.pack(); // C64Scr.setSize(380,300);

C64Win.setSize(386 \* 2 + 10, 284 \* 2 + 70);

C64Win.setResizable(true);

C64Win.setVisible(true);

KeyboardFocusManager.

getCurrentKeyboardFocusManager().addKeyEventDispatcher(this);

// Setup disk sounds

AudioClip trackSound = null;

AudioClip motorSound = null;

URL url = getClass().getResource("sounds/track.wav");

if (url != null) trackSound = Applet.newAudioClip(url);

url = getClass().getResource("sounds/motor.wav");

if (url != null) motorSound = Applet.newAudioClip(url);

scr.setSounds(trackSound, motorSound);

c64Canvas = (KeyListener) scr.getScreen();

}

private void createRadioMenu(JMenu subm, String[] names, int selected) {

ButtonGroup group = new ButtonGroup();

JRadioButtonMenuItem mi;

for (int i = 0; i < names.length; i++) {

subm.add(mi = new JRadioButtonMenuItem(names[i]));

if (i == selected)

mi.setSelected(true);

mi.addActionListener(this);

group.add(mi);

}

}

public boolean dispatchKeyEvent(KeyEvent e) {

if (C64Win.isFocused()) {

if (e.getID() == KeyEvent.KEY\_PRESSED) {

c64Canvas.keyPressed(e);

} else if (e.getID() == KeyEvent.KEY\_RELEASED) {

c64Canvas.keyReleased(e);

} else if (e.getID() == KeyEvent.KEY\_TYPED) {

c64Canvas.keyTyped(e);

}

return true;

}

return false;

}

public void actionPerformed(ActionEvent ae) {

String cmd = ae.getActionCommand();

if ("Open File/Disk".equals(cmd)) {

if (fileTable == null) {

// Show the table somewhere!!!

fileTable = new JTable(dataModel);

fileTable.getColumnModel().getColumn(1).setMaxWidth(50);

fileTable.getColumnModel().getColumn(2).setMaxWidth(50);

fileTable.setShowGrid(false);

fileTable.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

}

readDisk();

} else if ("Reset".equals(cmd)){

cpu.reset();

} else if ("Hard Reset".equals(cmd)){

cpu.hardReset();

} else if ("About JaC64".equals(cmd)) {

showAbout();

} else if ("Load File".equals(cmd)) {

if (loadFile == null) {

loadFile = new JDialog(C64Win, "Load file from disk");

loadFile.setAlwaysOnTop(true);

loadFile.setVisible(true);

loadFile.setLayout(new BorderLayout());

loadFile.add(new JScrollPane(fileTable), BorderLayout.CENTER);

loadFile.add(fileTable.getTableHeader(), BorderLayout.NORTH);

JPanel jp = new JPanel();

JButton jb;

jp.add(jb = new JButton("Load file"));

jb.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent ae) {

reader.readFile(dirEntries[fileTable.getSelectedRow()].name);

loadFile.setVisible(false);

}

});

jp.add(jb = new JButton("Cancel"));

jb.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent ae) {

loadFile.setVisible(false);

}

});

loadFile.add(jp, BorderLayout.SOUTH);

loadFile.setSize(300, 400);

}

loadFile.setVisible(true);

} else if (cmd.startsWith("Color Set")) {

int cs = cmd.charAt(10) - '1';

System.out.println("Color set: " + cs);

scr.setColorSet(cs);

} else if (cmd.equals(SID\_TYPES[0])) {

scr.setSID(C64Screen.RESID\_6581);

} else if (cmd.equals(SID\_TYPES[1])) {

scr.setSID(C64Screen.RESID\_8580);

} else if (cmd.equals(SID\_TYPES[2])) {

scr.setSID(C64Screen.JACSID);

} else if (cmd.equals(JOYSTICK[0])) {

scr.setStick(true);

} else if (cmd.equals(JOYSTICK[1])) {

scr.setStick(false);

}

}

private void showAbout() {

JOptionPane.showMessageDialog(C64Win, ABOUT\_MESSAGE,

"JaC64 - The Java C64 Emulator", JOptionPane.INFORMATION\_MESSAGE);

}

public void toggleFullScreen() {

System.out.println("Toggle fullscreen called!");

setFull(!fullscreen);

}

private void readDisk() {

if (fileDialog == null)

fileDialog = new FileDialog(C64Win, "Select File/Disk to Load");

fileDialog.setVisible(true);

String name = fileDialog.getDirectory() + fileDialog.getFile();

if (!readDisk(name)) {

dirEntries = (DirEntry[]) reader.getDirNames().toArray(new DirEntry[0]);

fileTable.tableChanged(new TableModelEvent(dataModel));

}

}

private boolean readDisk(String name) {

System.out.println("READING FROM: " + name);

if ((name.toLowerCase()).endsWith(".d64"))

reader.readDiskFromFile(name);

else if ((name.toLowerCase()).endsWith(".t64"))

reader.readTapeFromFile(name);

else if (name.toLowerCase().endsWith(".prg") ||

name.toLowerCase().endsWith(".p00")) {

cpu.reset();

try {

Thread.sleep(10);

}catch (Exception e2) {

System.out.println("Exception while sleeping...");

}

while(!scr.ready()) {

try {

Thread.sleep(100);

}catch (Exception e2) {

System.out.println("Exception while sleeping...");

}

}

reader.readPGM(name, -1);

cpu.runBasic();

return true;

}

return false;

}

private boolean readDisk(URL url) {

String name = url.toString();

System.out.println("READING FROM URL: " + name);

if ((name.toLowerCase()).endsWith(".d64"))

reader.readDiskFromURL(url);

else if ((name.toLowerCase()).endsWith(".t64"))

reader.readTapeFromURL(url);

else if (name.toLowerCase().endsWith(".prg") ||

name.toLowerCase().endsWith(".p00")) {

cpu.reset();

try {

Thread.sleep(10);

}catch (Exception e2) {

System.out.println("Exception while sleeping...");

}

while(!scr.ready()) {

try {

Thread.sleep(100);

}catch (Exception e2) {

System.out.println("Exception while sleeping...");

}

}

reader.readPGM(url, -1);

cpu.runBasic();

return true;

}

return false;

}

private void setFull(boolean full) {

// JWindow jw = full ? C64Scr : null;

// java.awt.GraphicsEnvironment.getLocalGraphicsEnvironment().

// getDefaultScreenDevice().setFullScreenWindow(jw);

// if (!full) {

// C64Scr.setSize(386 \* 2, 284 \* 2);

// C64Scr.validate();

// }

// fullscreen = full;

}

private void waitForKernal() {

while(!scr.ready()) {

try {

Thread.sleep(100);

} catch (Exception e2) {

System.out.println("Exception while sleeping...");

}

}

}

private void autoStart(String filename) {

Thread t = new Thread(new Runnable() {

public void run() {

waitForKernal();

System.out.println("Kernal READY!");

URL url = getClass().getResource(filename);

if (url != null) {

readDisk(url);

} else {

readDisk(filename);

}

}

});

t.start();

}

public static void main(String[] args) {

String autostart = null;

for (int i = 0; i < args.length; i++) {

if (args[i].equals("-a")) {

i++;

autostart = args[i];

} else {

System.out.println("Usage: java [-cp <classpath>] JaC64 [-a <autostart(.d64|.t64|.prg|.p00)>]");

System.exit(1);

}

}

JaC64 emu = new JaC64();

if (autostart != null) {

emu.autoStart(autostart);

}

emu.cpu.start();

}

}

