package lte;

import java.io.IOException;

import java.net.URL;

import java.util.ResourceBundle;

import javafx.animation.KeyFrame;

import javafx.animation.Timeline;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.fxml.Initializable;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.ChoiceBox;

import javafx.scene.control.Label;

import javafx.scene.control.TableColumn;

import javafx.scene.control.TableView;

import javafx.scene.control.TextArea;

import javafx.scene.control.cell.PropertyValueFactory;

import javafx.scene.input.MouseEvent;

import javafx.stage.Stage;

import javafx.stage.StageStyle;

import javafx.util.Duration;

public class LocalServingGatewayController implements Initializable {

double dragX = 0;

double dragY = 0;

int coun=0,coun1=1;

public static String svideoname;

@FXML

public Label gatewayn;

@FXML

public ChoiceBox countChoiceBox;

@FXML

public Label strlvideo;

@FXML

public TextArea localvideo;

@FXML

public TextArea reqsource;

@FXML

public Label strtraffic;

@FXML

public Label traffic;

@FXML

public Label port;

@FXML

public Label strport;

@FXML

public Label ram;

@FXML

public Label strram;

@FXML

public Label memory;

@FXML

public Label strmemory;

public ObservableList<Destsplit> dpathob;

@FXML

public TableView<Destsplit> dflowpath;

@FXML

public TableColumn<Destsplit, String> Columndestination;

@FXML

public TableColumn<Destsplit, String> Columnnext;

@FXML

public TableColumn<Destsplit, String> Columnsource;

RandomClass ran = new RandomClass();

Observer o = new Observer();

Gatewaymultireceiver gmr;

Gatewaysinglereceiver gsr;

Gatewaymultisender gms;

String portno = ran.portn();

String sysno = ran.sysname();

@FXML

private void send(ActionEvent event) throws IOException

{

// gsr.send();

}

String value,travalue,ramvalue,memvalue;

@Override

public void initialize(URL url, ResourceBundle rb)

{

countChoiceBox.getItems().addAll("LOW","HIGH");

strport.setText(portno);

value = countChoiceBox.getValue().toString();

memvalue = ran.trafficlevel();

strmemory.setText(memvalue);

ramvalue = ran.trafficlevel();

strram.setText(ramvalue);

travalue = ran.trafficlevel();

traffic.setText(travalue);

gatewayn.setText(FXMLDocumentController.gatewayname);

final Timeline timeline2 = new Timeline();

timeline2.setCycleCount(Timeline.INDEFINITE);

timeline2.getKeyFrames().add(new KeyFrame(Duration.seconds(3),new EventHandler<ActionEvent>()

{

@Override

public void handle(ActionEvent event)

{

String s = gmr.urlcall();

String value = countChoiceBox.getValue().toString();

if(value.equalsIgnoreCase("High"))

{

if(coun==0)

{

travalue=ran.trafficllevel();

ramvalue = ran.trafficllevel();

strram.setText(ramvalue);

traffic.setText(travalue);

coun=coun+1;

coun1=0;

}

}

if(value.equalsIgnoreCase("low"))

{

if(coun1==0)

{

travalue=ran.trafficlevel();

ramvalue = ran.trafficlevel();

strram.setText(ramvalue);

traffic.setText(travalue);

coun1++;

coun=0;

}

}

gms = new Gatewaymultisender(gatewayn.getText(), portno, sysno,value,travalue);

}

},null));

timeline2.play();

final Timeline timeline3 = new Timeline();

timeline3.setCycleCount(Timeline.INDEFINITE);

timeline3.getKeyFrames().add(new KeyFrame(Duration.seconds(90),new EventHandler<ActionEvent>()

{

@Override

public void handle(ActionEvent event)

{

travalue = ran.trafficlevel();

traffic.setText(travalue);

if(countChoiceBox.getValue().toString().equalsIgnoreCase("high"))

{

travalue = ran.trafficllevel();

traffic.setText(travalue);

}

else if(countChoiceBox.getValue().toString().equalsIgnoreCase("low"))

{

travalue = ran.trafficlevel();

traffic.setText(travalue);

}

gms = new Gatewaymultisender(gatewayn.getText(),travalue);

}

},null));

timeline3.play();

gmr = new Gatewaymultireceiver(gatewayn.getText(), portno, sysno);

gsr = new Gatewaysinglereceiver(gatewayn.getText(),portno,sysno,gmr,this);

localvideo.textProperty().bindBidirectional(o.getteravailablevideo());

reqsource.textProperty().bindBidirectional(o.getterreqsource());

dpathob = FXCollections.observableArrayList(new Destsplit());

Columndestination.setCellValueFactory(new PropertyValueFactory<Destsplit, String>("Columndestination"));

Columnnext.setCellValueFactory(new PropertyValueFactory<Destsplit, String>("Columnnext"));

Columnsource.setCellValueFactory(new PropertyValueFactory<Destsplit, String>("Columnsource"));

dflowpath.setItems(dpathob);

dflowpath.getColumns().setAll(Columndestination, Columnnext, Columnsource);

}

public void createnode(String videoname) throws IOException

{

svideoname = videoname;

Parent root = FXMLLoader.load(getClass().getResource("SuperSource.fxml"));

Stage stage = new Stage();

stage.initStyle(StageStyle.TRANSPARENT);

Scene scene = new Scene(root);

stage.setScene(scene);

stage.show();

root.setOnMouseDragged(new EventHandler<MouseEvent>()

{

public void handle(MouseEvent me)

{

stage.setX(me.getScreenX() - dragX);

stage.setY(me.getScreenY() - dragY);

}

});

root.setOnMousePressed(new EventHandler<MouseEvent>()

{

public void handle(MouseEvent me)

{

dragX = me.getScreenX() - stage.getX();

dragY = me.getScreenY() - stage.getY();

}

});

}

public Observer getObeserver()

{

return o;