СПб НИУ ИТМО

кафедра ИПМ

Программирование интернет-приложений

Лабораторная работа № 9

JavaServer Faces Framework

Вариант 21274

Работу выполнил:

Студент II курса

Группы № 2120

Журавлев Виталий

Санкт-Петербург

2014 г.

**Цель работы:**

Разработать приложение на базе JavaServer Faces Framework, которое осуществляет проверку попадания точки в заданную область на координатной плоскости.

Приложение должно включать в себя 2 facelets-шаблона - стартовую страницу и основную страницу приложения, а также набор управляемых бинов (managed beans), реализующих логику на стороне сервера.

**Стартовая страница должна содержать следующие элементы:**

* "Шапку", содержащую ФИО студента, номер группы и номер варианта.
* Интерактивные часы, показывающие текущие дату и время, обновляющиеся раз в 12 секунд.
* Ссылку, позволяющую перейти на основную страницу приложения.

**Основная страница приложения должна содержать следующие элементы:**

* Набор компонентов для задания координат точки и радиуса области в соответствии с вариантом задания. Может потребоваться использование дополнительных библиотек компонентов - ICEfaces (префикс "ace") и PrimeFaces (префикс "p"). Если компонент допускает ввод заведомо некорректных данных (таких, например, как буквы в координатах точки или отрицательный радиус), то приложение должно осуществлять их валидацию.
* Динамически обновляемую картинку, изображающую область на координатной плоскости в соответствии с номером варианта и точки, координаты которых были заданы пользователем. Клик по картинке должен инициировать сценарий, осуществляющий определение координат новой точки и отправку их на сервер для проверки её попадания в область. Цвет точек должен зависить от факта попадания / непопадания в область. Смена радиуса также должна инициировать перерисовку картинки.
* Таблицу со списком результатов предыдущих проверок.
* Ссылку, позволяющую вернуться на стартовую страницу.

**Дополнительные требования к приложению:**

* Для хранения списка результатов должен использоваться Application-scoped Managed Bean.
* Конфигурация управляемых бинов должна быть задана с помощью параметров в конфигурационном файле.
* Правила навигации между страницами приложения должны быть заданы в отдельном конфигурационном файле.

****

**Код Программы:**

Index.xhtml:

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml"

 xmlns:h="http://java.sun.com/jsf/html"

 xmlns:ui="http://java.sun.com/jsf/facelets"

 xmlns:p="http://primefaces.org/ui">

 <h:head>

 <title>Welcome to Lab9</title>

 </h:head>

 <h:body style=" background-color:lightgoldenrodyellow;">

 <h1 align="center" style="color:darkgreen; padding: 3px">

 Журавлев Виталий, гр.2120 <br/>

 Вариант 21274

 </h1>

 <p style="text-align: center">

 <p:clock pattern="dd MMMM hh:mm:ss" syncInterval="12000" autoSync="true" mode="client"/>

 </p>

 <h:form>

 <p style="text-align: center">

 <h:commandButton value="Start!" action="#{NavigationBean.main}" style="width: 120px"/>

 </p>

 </h:form>

 </h:body>

</html>

Main.xhml:

<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml"

 xmlns:h="http://java.sun.com/jsf/html"

 xmlns:f="http://java.sun.com/jsf/core"

 xmlns:p="http://primefaces.org/ui">

 <h:head>

 <title>Lab9 var.21274</title>

 <script src="/Lab9/check.js" type="text/javascript" />

 </h:head>

 <h:body>

 <h:form id="mainForm" style="padding-left: 15px; padding-top: 15px; padding-bottom: 10px">

 X:

 <h:selectOneMenu id="sXVal">

 <f:selectItem id="item1" itemLabel="-5" itemValue="-5.0" />

 <f:selectItem id="item2" itemLabel="-4" itemValue="-4.0" />

 <f:selectItem id="item3" itemLabel="-3" itemValue="-3.0" />

 <f:selectItem id="item4" itemLabel="-2" itemValue="-2.0" />

 <f:selectItem id="item5" itemLabel="-1" itemValue="-1.0" />

 <f:selectItem id="item6" itemLabel="0" itemValue="0.0" />

 <f:selectItem id="item7" itemLabel="1" itemValue="1.0" />

 <f:selectItem id="item8" itemLabel="2" itemValue="2.0" />

 <f:selectItem id="item9" itemLabel="3" itemValue="3.0" />

 </h:selectOneMenu>

 <h:inputHidden id="XVal" value="#{ControllerBean.currentX}"/>

 <h:outputText value="#{ControllerBean.currentX}"/>

 <br/>

 Y:

 <h:inputText id="YVal" value="#{ControllerBean.currentY}">

 </h:inputText>

 <br/>

 R:

 <h:inputHidden id="RVal" value="#{ControllerBean.currentR}"/>

 <h:outputText value="#{ControllerBean.currentR/10}"/>

 <p:slider onSlideEnd="resetR()" for="RVal" minValue="20" maxValue="50" style="width: 120px"/>

 <br/>

 <h:commandButton value="Add Point" action="#{ControllerBean.checkAndSubmit2()}" style="width: 120px" />

 <h:commandLink action="#{ControllerBean.checkAndSubmit2()}" style="display:none;" id="hiddenSubmit"/>

 <h:commandLink action="#{ControllerBean.checkAndSubmit()}" style="display:none;" id="hiddenSubmit2"/>

 <br/>

 <h:commandButton value="Clear" action="#{ControllerBean.clearHistory()}" style="width: 120px"/>

 <br/><br/>

 <h:commandButton id="Back" value="Back" action="#{NavigationBean.index}" style="width: 120px" />

 </h:form>

 <h:inputHidden id="Rval"/>

 <p:graphicImage onclick="imgClickHandler(event)" id="image" value="#{ViewBean.image}" cache="false" style="position: absolute; right: 650px; top: 10px"/>

 <h:outputText value="#{ViewBean.htmlTable}" escape="false" style="position: absolute; right: 400px; top: 10px"/>

 </h:body>

</html>

Check.js:

function checkNumberValidation(number) {

 validNumber = "";

 pointFlag = false;

 desFlag = false;

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

 if ((number.charAt(i).charCodeAt(0) >= "0".charCodeAt(0)) && (number.charAt(i).charCodeAt(0) <= "9".charCodeAt(0)))

 validNumber += number.charAt(i);

 else if (((number.charAt(i) == ".")) && !pointFlag && (i != 0) && (i != 1 || !desFlag)) {

 pointFlag = true;

 validNumber += number.charAt(i);

 } else if ((i == 0) && (number.charAt(i) == "-")) {

 desFlag = true;

 validNumber += number.charAt(i);

 }

 }

 return validNumber;

}

function checkIntervalY(number)

{

 var value = number;

 if (value < -3.0 || value > 3.0)

 {

 window.alert("Y [-3;3]");

 value = 0;

 }

 return value;

}

function resetR()

{

 document.getElementById('mainForm:hiddenSubmit2').click();

}

function imgClickHandler(ClickEvent)

{

 var parrentPos = getPosition(ClickEvent.currentTarget);

 var xPos = ClickEvent.clientX - parrentPos.x;

 var yPos = ClickEvent.clientY - parrentPos.y;

 var height = ClickEvent.currentTarget.clientHeight;

 var width = ClickEvent.currentTarget.clientWidth;

 r = document.getElementById('Rval').value;

 var yVal = height / 2 - yPos;

 var xVal = xPos - width / 2;

 yVal = (10 \* yVal) / height;

 xVal = (10 \* xVal) / width;

 document.getElementById('mainForm:XVal').value = xVal;

 document.getElementById('mainForm:YVal').value = yVal;

 document.getElementById('mainForm:hiddenSubmit').click();

}

function getPosition(element)

{

 var xPos = 0;

 var yPos = 0;

 while (element)

 {

 xPos += (element.offsetLeft - element.scrollLeft + element.clientLeft);

 yPos += (element.offsetTop - element.scrollTop + element.clientTop);

 element = element.offsetParent;

 }

 return {x: xPos, y: yPos};

}

window.onload = function() {

 document.getElementById('mainForm:sXVal').onclick = function() {

 document.getElementById('mainForm:XVal').value = this.value;

 }

 document.getElementById('mainForm:YVal').onkeyup = function() {

 this.value = checkNumberValidation(this.value);

 }

 document.getElementById('mainForm:YVal').onchange = function() {

 this.value = checkIntervalY(this.value);

 }

}

Faces-config.xml:

<?xml version='1.0' encoding='UTF-8'?>

<faces-config version="2.2"

 xmlns="http://xmlns.jcp.org/xml/ns/javaee"

 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

 xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-facesconfig\_2\_2.xsd">

 <application>

 <managed-bean>

 <managed-bean-name>AreaBean</managed-bean-name>

 <managed-bean-class>com.device.AreaBean</managed-bean-class>

 <managed-bean-scope>application</managed-bean-scope>

 </managed-bean>

 <managed-bean>

 <managed-bean-name>ControllerBean</managed-bean-name>

 <managed-bean-class>com.device.ControllerBean</managed-bean-class>

 <managed-bean-scope>application</managed-bean-scope>

 </managed-bean>

 <managed-bean>

 <managed-bean-name>HistoryBean</managed-bean-name>

 <managed-bean-class>com.device.HistoryBean</managed-bean-class>

 <managed-bean-scope>application</managed-bean-scope>

 </managed-bean>

 <managed-bean>

 <managed-bean-name>NavigationBean</managed-bean-name>

 <managed-bean-class>com.device.NavigationBean</managed-bean-class>

 <managed-bean-scope>application</managed-bean-scope>

 </managed-bean>

 <managed-bean>

 <managed-bean-name>ViewBean</managed-bean-name>

 <managed-bean-class>com.device.ViewBean</managed-bean-class>

 <managed-bean-scope>application</managed-bean-scope>

 </managed-bean>

 <navigation-rule>

 <navigation-case>

 <from-outcome>index</from-outcome>

 <to-view-id>/index.xhtml</to-view-id>

 </navigation-case>

 <navigation-case>

 <from-outcome>main</from-outcome>

 <to-view-id>/main.xhtml</to-view-id>

 </navigation-case>

 </navigation-rule>

 </application>

</faces-config>

Classes:

public class **AreaBean** implements Serializable {

 public HistoryItem solve(double X, double Y, double R) {

 HistoryItem item = new HistoryItem(X, Y, R, checkArea(X, Y, R));

 return item;

 }

 boolean checkArea(double X, double Y, double R) {

 return (((X <= 0) && (Y <= 0) && (Math.pow(X, 2) + Math.pow(Y, 2) <= Math.pow(R, 2)))

 || ((X <= 0) && (Y >= 0) && (X >= -R / 2) && (Y <= R))

 || ((X >= 0) && (Y >= 0) && (Y <= R / 2 - X)));

 }

}

public class **ControllerBean** implements Serializable {

 private HistoryBean history;

 private AreaBean area;

 private double currentR;

 private double currentY;

 private double currentX;

 private int pointCount = 0;

 @PostConstruct

 public void init() {

 setCurrentR(30);

 }

 public void setHistory(HistoryBean history) {

 this.history = history;

 }

 public void setArea(AreaBean area) {

 this.area = area;

 }

 public double getCurrentR() {

 return currentR;

 }

 public double getCurrentY() {

 return currentY;

 }

 public double getCurrentX() {

 return currentX;

 }

 public void setCurrentR(double currentR) {

 this.currentR = currentR;

 }

 public void setCurrentY(double currentY) {

 this.currentY = currentY;

 }

 public void setCurrentX(double currentX) {

 this.currentX = currentX;

 }

 public void clearHistory() {

 this.pointCount = 0;

 this.history.Clear();

 }

 public void checkAndSubmit() {

 HistoryItem item = area.solve(currentX, currentY, currentR / 10);

 if (item != null) {

 history.addItem(item);

 }

 pointCount++;

 }

 public void checkAndSubmit2() {

 HistoryItem[] tempArr = history.toArray();

 if (pointCount > 1) {

 for (int i = tempArr.length - (pointCount); i < tempArr.length; i++) {

 HistoryItem item = area.solve(tempArr[i].getX(), tempArr[i].getY(), currentR / 10);

 if (item != null) {

 history.addItem(item);

 }

 }

 }

 }

}

public class **HistoryBean** implements Serializable {

 Vector<HistoryItem> history;

 @PostConstruct

 public void init() {

 history = new Vector<HistoryItem>();

 }

 public void addItem(HistoryItem item) {

 history.add(item);

 }

 public HistoryItem[] toArray() {

 return history.toArray(new HistoryItem[history.size()]);

 }

 public void Clear() {

 if(history.toArray().length >= 1)

 history.clear();

 }

}

public class **HistoryItem** {

 double x;

 double y;

 double r;

 boolean result;

 public HistoryItem(double lx,double ly, double lr, boolean lresult) {

 this.x = lx;

 this.y = ly;

 this.r = lr;

 this.result = lresult;

 }

 public double getX() {

 return x;

 }

 public double getY() {

 return y;

 }

 public double getR() {

 return r;

 }

 public boolean getResult() {

 return result;

 }

 public String toString() {

 return "X: " + x + "\n Y: " + y;

 }

}

public class **ImgGenerator** {

 private static final double ImageResolutionR = 10;

 private static final int PointRadius = 15;

 public static BufferedImage genImage(HistoryItem[] items, double width, double height, double R) {

 BufferedImage result;

 int r\_px;

 r\_px = (int) ((R \* height) / ImageResolutionR);

 result = new BufferedImage((int) width, (int) height, BufferedImage.TYPE\_INT\_RGB);

 Graphics g = result.createGraphics();

 g.setColor(Color.WHITE);

 g.fillRect(0, 0, (int) width, (int) height);

 paintArea(g, (int) width, (int) height, r\_px);

 paintAxes(g, (int) width, (int) height, r\_px);

 paintPoints(g, items, (int) width, (int) height, R);

 return result;

 }

 static void paintAxes(Graphics g, int width, int height, int R) {

 int XCenter = width / 2;

 int YCenter = height / 2;

 int HalfR = R / 2;

 g.setColor(Color.BLACK);

 g.drawLine(0, YCenter, 2 \* XCenter, YCenter);

 g.drawLine(XCenter, 0, XCenter, 2 \* YCenter);

 g.drawLine(XCenter - R, YCenter + 5, XCenter - R, YCenter - 5);

 g.drawLine(XCenter + R, YCenter - 5, XCenter + R, YCenter + 5);

 g.drawLine(XCenter + 5, YCenter + R, XCenter - 5, YCenter + R);

 g.drawLine(XCenter + 5, YCenter - R, XCenter - 5, YCenter - R);

 g.drawLine(XCenter - HalfR, YCenter + 5, XCenter - HalfR, YCenter - 5);

 g.drawLine(XCenter + HalfR, YCenter - 5, XCenter + HalfR, YCenter + 5);

 g.drawLine(XCenter + 5, YCenter + HalfR, XCenter - 5, YCenter + HalfR);

 g.drawLine(XCenter + 5, YCenter - HalfR, XCenter - 5, YCenter - HalfR);

 }

 static void paintArea(Graphics g, int width, int height, int R) {

 int xc = width / 2;

 int yc = height / 2;

 g.setColor(Color.BLUE);

 g.fillArc(xc - R, yc - R, 2 \* R, 2 \* R, -90, -90);

 g.fillRect(xc - R / 2, yc - R, R / 2, R);

 g.fillPolygon(getTrianglePolygon(width, height, R));

 }

 static void paintPoints(Graphics g, HistoryItem[] items, int width, int height, double R) {

 for (HistoryItem item : items) {

 Color color = ((item.getResult()) ? Color.GREEN : Color.RED);

 g.setColor(color);

 int item\_y, item\_x;

 item\_y = (int) ((item.getY() \* height) / ImageResolutionR) + 7;

 item\_x = (int) ((item.getX() \* height) / ImageResolutionR) - 7;

 item\_x = width / 2 + item\_x;

 item\_y = height / 2 - item\_y;

 g.setColor(color);

 g.fillOval(item\_x, item\_y, PointRadius, PointRadius);

 }

 }

 static Polygon getTrianglePolygon(int width, int height, int R) {

 int num = 3;

 int[] x\_points = new int[num];

 int[] y\_points = new int[num];

 x\_points[0] = width / 2;

 y\_points[0] = height / 2;

 x\_points[1] = width / 2 + R/2;

 y\_points[1] = height / 2;

 x\_points[2] = width / 2;

 y\_points[2] = height / 2 - R/2;

 Polygon result = new Polygon(x\_points, y\_points, num);

 return result;

 }

}

public class **ViewBean** implements Serializable {

 private HistoryBean historyBean;

 private StreamedContent image;

 private ControllerBean controllerBean;

 public void setControllerBean(ControllerBean controllerBean) {

 this.controllerBean = controllerBean;

 }

 private String htmlTable;

 private double imageWidht;

 private double imageHeight;

 private static final String inAreaMSG = "True";

 private static final String outAreaMSG = "False";

 private static final String TableFirstSTR = "<tr><td width=\"25%\">X</td><td width=\"25%\">Y</td>"

 + "<td width=\"25%\">R</td><td width=\"25%\">Result</td></tr>";

 @PostConstruct

 public void init() {

 imageWidht = 500;

 imageHeight = 500;

 }

 public String getHtmlTable() {

 htmlTable = genHTMLTable();

 return htmlTable;

 }

 public void setImage(StreamedContent image) {

 this.image = image;

 }

 public StreamedContent getImage() {

 setImage(gnImage());

 return image;

 }

 public void setHistoryBean(HistoryBean historyBean) {

 this.historyBean = historyBean;

 }

 public void setImageWidht(double imageWidht) {

 this.imageWidht = imageWidht;

 }

 public void setImageHeight(double imageHeight) {

 this.imageHeight = imageHeight;

 }

 StreamedContent gnImage() {

 DefaultStreamedContent result = null;

 try {

 ByteArrayOutputStream os = new ByteArrayOutputStream();

 BufferedImage img = ImgGenerator.genImage(historyBean.toArray(), imageWidht, imageHeight, controllerBean.getCurrentR() / 10);

 ImageIO.write(img, "png", os);

 result = new DefaultStreamedContent(new ByteArrayInputStream(os.toByteArray()), "image/png");

 } catch (IOException ex) {

 }

 return result;

 }

 String genHTMLTable() {

 StringBuilder build = new StringBuilder();

 build.append("<table border=\"2\">");

 build.append(TableFirstSTR);

 for (HistoryItem item : historyBean.toArray()) {

 build.append("<tr>");

 build.append("<td width=\"25%\">").append(item.getX()).append("</td><td width=\"25%\">").append(item.getY()).append("</td><td width=\"25%\">").append(item.getR()).append("</td>");

 build.append("<td width=\"25%\">").append(item.getResult() ? inAreaMSG : utAreaMSG).append("</td>");

 build.append("</tr>");

 }

 build.append("</table>");

 return build.toString();

 }

}

**Вывод:**

В ходе выполнения лабораторной работы я изучил некоторые правила работы c JSF-компонентами: синтаксис, конфигурационные файлы, управляемые бины, написание кода в формате xhtml с использованием Facelets-шаблонов.

 В том числе я освоил такое понятие, как шаблоны проектирования и архитектурные шаблоны.

Так же я познакомился с работой с графиками, их рисованием и отображением точек на рисунке и обработкой полученных значений в виде таблицы истории результатов.