САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Кафедра Вычислительной техники

Лабораторная работа №9

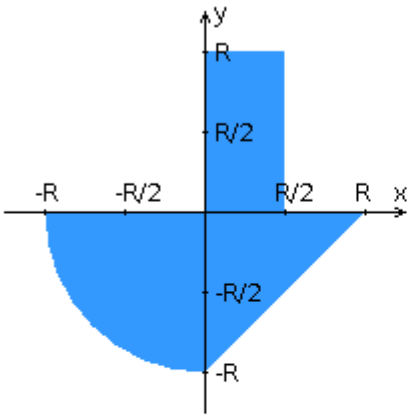
Выполнил:
студент II курса группы 2125
Припадчев Артём

Проверит:
Харитонова А.Е.

Санкт-Петербург
2014

Задание: 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.



изменение X: inputText (-3 ... 5)

изменение Y: selectOneRadio {'-4','-3','-2','-1','0','1','2','3','4'}

изменение R: commandButton {'1','2','3','4', '5'}

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 5 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 аннотаций.
- Правила навигации между страницами приложения должны быть заданы в отдельном конфигурационном файле.

Исходный ход

Scripts

check.js

```
 function isCharNumber(charNumber){
if (( charNumber.charCodeAt(0) >= "0".charCodeAt(0) ) && (charNumber.charCodeAt(0) <=
"9".charCodeAt(0))) return true;
return false;
}

function checkNumberValidation(number) {
validNumber = "";
pointFlag = false;
desFlag=false;

for (i = 0 ; i < number.length ; i++) {
if ( isCharNumber(number.charAt(i)) ) 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 checkIntervalX(number)
{
   var value = number;
   if(value < -3.0 || value >5.0)
   {
      window.alert("X [-3;5]");
      value = 0;
   }
   return value;
}

function checkIntervalY(number)
{
   var value = number;
   if(value < -4.0 || value >4.0)
   {
      window.alert("Y [-4;4]");
      value = 0;
   }
   return value;
}

window.onload = function() {
document.getElementById('mainForm:XVal').onkeyup=function(){
this.value = checkNumberValidation(this.value);
```

```
}
document.getElementById('mainForm:YVal').onkeyup=function(){
this.value = checkNumberValidation(this.value);
}
document.getElementById('mainForm:XVal').onchange=function(){
this.value = checkIntervalX(this.value);
}
document.getElementById('mainForm:YVal').onchange=function(){
this.value = checkIntervalY(this.value);
}
}
```

img_proc.js

```
 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;
   // xScaled = 2*(xPos - height/2)/height;
 //   yScaled = -2*(yPos - width/2)/width;


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

//    var yVal = yScaled*r;
//    var xVal = xScaled*r;


    var yVal = height/2 - yPos;
    var xVal = xPos - width/2;


    yVal = (8*yVal)/height;
    xVal = (8*xVal)/width;

//    alert(xVal + " " + yVal + '\n' + r);

    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};
}
```

Страницы

index_template.xhtml

```
 <?xml version="1.0" encoding="UTF8"?>
<!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" >

<h:head>

</h:head>
<h:body>
    <div id="header">
        <ui:insert name="header">

        </ui:insert>
    </div>

    <div id="clock">
        <ui:insert name="clock">

        </ui:insert>
    </div>

    <h:form>
        <h:commandButton id="MainPage" value="Main Page"
                         action="#{NavigationBean.main}" />
    </h:form>
</h:body>
</html>
```

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>Index page</title>
</h:head>
<h:body>
    <ui:composition template="/templates/index_template.xhtml">
        <ui:define name="header">
```

```
            <p>
            Припадчев Артём, гр.2125 <br/>
            Вариант 21262
            </p>
        </ui:define>

        <ui:define name="clock">
            <p>
            <div id="clock">
                <p:clock autoSync="false" syncInterval="5000000" mode="client"/>
            </div>
            </p>
        </ui:define>
    </ui:composition>
</h:body>
</html>
```

main_template.xhtml

```
 <?xml version="1.0" encoding="UTF8" ?>
<!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"
      xmlns:f="http://xmlns.jcp.org/jsf/core">
<h:head>
    <title>Main page</title>
    <script src="/LastLab/scripts/img_proc.js" type="text/javascript" language="JavaScript" ></script>
    <script src="/LastLab/scripts/check.js" type="text/javascript" />
</h:head>
<h:body>
    <div id="control"  style="padding-left: 15px; padding-top: 15px; padding-bottom: 10px">
        <ui:insert name="controls">
            Controls will be here
        </ui:insert>
    </div>

    <div id="results" style="position: absolute; right: 650px; top: 10px">
        <ui:insert name="results">
            Results will be here
        </ui:insert>
    </div>
    <div id="list" style="position: absolute; right: 400px; top: 10px">
        <ui:insert name="list">
            List will be here
        </ui:insert>
    </div>

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

```
      <h:commandButton id="Back" value="Back"
                    action="#{NavigationBean.index}" style="width: 60px" />
      <h:commandButton value="Clear" action="#{ControllerBean.clearHistory()}" style="width: 60px">
        </h:commandButton>
   </h:form>
</h:body>
</html>
```

main.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:f="http://java.sun.com/jsf/core"
     xmlns:p="http://primefaces.org/ui">
<h:head>
   <title>Main page</title>
   <script src="/LastLab/scripts/img_proc.js" type="text/javascript" />
   <script src="/LastLab/scripts/check.js" type="text/javascript" />
</h:head>
<h:body>
   <ui:composition template="main_template.xhtml" >
      <ui:define name="controls">
         <h:form id="mainForm">
            Enter X: <h:inputText id="XVal" value="#{ControllerBean.currentX}" >
             <f:validateLongRange minimum="-3" maximum="5"/>
         </h:inputText>
         <br/>
         Enter Y: <h:inputText id="YVal" value="#{ControllerBean.currentY}"  >
                  <f:validateLongRange minimum="-4" maximum="4"/>
               </h:inputText>
            <br />
         Choose R:<br/>

         <h:commandButton value="1" action="#{ControllerBean.checkAndSubmit()}" style="width:
60px">
            <f:setPropertyActionListener target="#{ControllerBean.currentR}" value="1"/>
         </h:commandButton><br/>
         <h:commandButton value="1.5" action="#{ControllerBean.checkAndSubmit()}" style="width:
60px">
            <f:setPropertyActionListener target="#{ControllerBean.currentR}" value="1.5"/>
         </h:commandButton><br/>
         <h:commandButton value="2" action="#{ControllerBean.checkAndSubmit()}" style="width:
60px">
            <f:setPropertyActionListener target="#{ControllerBean.currentR}" value="2"/>
         </h:commandButton><br/>
         <h:commandButton value="2.5" action="#{ControllerBean.checkAndSubmit()}" style="width:
60px">
            <f:setPropertyActionListener target="#{ControllerBean.currentR}" value="2.5"/>
         </h:commandButton><br/>
         <h:commandButton value="3" action="#{ControllerBean.checkAndSubmit()}" style="width:
60px">
```

```
            <f:setPropertyActionListener target="#{ControllerBean.currentR}" value="3"/>
         </h:commandButton><br/>
            <h:commandLink action="#{ControllerBean.checkAndSubmit2()}" style="display:none;"
id="hiddenSubmit">
            </h:commandLink>
         </h:form>
         <h:inputHidden id="Rval" value="#{ControllerBean.currentR}"></h:inputHidden>
      </ui:define>
      <ui:define name="results">
         <p:graphicImage onclick="imgClickHandler(event)" id="image" value="#{ViewBean.image}"
cache="false"/>
      </ui:define>
      <ui:define name="list">
        <h:outputText value="#{ViewBean.htmlTable}" escape="false" />
      </ui:define>

   </ui:composition>
</h:body>
</html>
```

AreaBean.java

```
 package Beans;

import javax.annotation.PostConstruct;
import javax.faces.bean.ApplicationScoped;
import javax.faces.bean.ManagedBean;
import java.io.Serializable;

@ManagedBean(name="AreaBean")
@ApplicationScoped
public class AreaBean implements Serializable
{
   private static double XMinLimit = -5;
   private static double XMaxLimit = 5;
   private static double YMaxLimit = -5;
   private static double YMinLimit = 3;


   public HistoryItem solve(double X, double Y, double R)
   {
      HistoryItem item = null;
      //if(checkValues(X,Y,R))
         item = new HistoryItem(X,Y,R, checkArea(X,Y,R));

      return item;
   }

   boolean checkValues(double X, double Y, double R)
   {
      boolean result = true;
      if(X < XMinLimit || X > XMaxLimit)
         result = false;
      if(Y < YMinLimit || Y > YMaxLimit)
```

```
        result = false;

    return result;
}

boolean checkArea(double X, double Y, double R)
{
    boolean res = false;

    if(X > 0)
    {
       if(Y < 0)
          res = FourthQuarter(X, Y, R);
       else
       {
          res = FirstQuarter(X,Y,R);
       }
    }
    else
    {
        if(Y < 0)
            res = ThirdQuarter(X, Y, R);
    }
   return res;
}

boolean FirstQuarter(double X, double Y, double R)
{
    boolean res = false;

    if(X < R/2 && Y < R)
    {
       res = true;
    }
    return res;
}

boolean ThirdQuarter(double X, double Y, double R)
{
    boolean res = false;

    if(Math.pow(X,2) + Math.pow(Y, 2) < Math.pow(R,2))
       res = true;

    return res;
}

boolean FourthQuarter(double X, double Y, double R)
{
    boolean res = false;

    if(X/2 - R/2 > Y/2)
```

```
            res = true;
        return !res;
    }
}
```

ControllerBean.java

```
 package Beans;

import javax.annotation.PostConstruct;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.ApplicationScoped;
import javax.faces.bean.ManagedProperty;
import java.io.Serializable;

@ManagedBean(name="ControllerBean",eager = true)
@ApplicationScoped
public class ControllerBean implements Serializable
{
    @ManagedProperty(value="#{HistoryBean}")
    private HistoryBean history;
    @ManagedProperty(value="#{AreaBean}")
    private AreaBean    area;
   /* @ManagedProperty(value="#{ViewBean}")
    private ViewBean view;
*/

    @PostConstruct
    public void init()
    {
        setCurrentR(3f);
    }
/*
    public void setView(ViewBean view) {
        this.view = view;
    }*/

    private double currentR;
    private int pointCount = 0;

    public void setHistory(HistoryBean history)
    {
        this.history = history;
    }

    public void setArea(AreaBean area) {
        this.area = area;
    }

    private double currentY;
    private double currentX;
```

```
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 checkAndSubmit2()
{
    HistoryItem item = area.solve(currentX, currentY, currentR);
    if(item != null)
        history.addItem(item);
    pointCount++;
    //checkAndSubmit();
}
public void checkAndSubmit()
{
    HistoryItem[] tempArr = history.toArray();
    if(pointCount > 0)
    {
        for (int i = tempArr.length - (pointCount);i<tempArr.length;i++)
        {
            HistoryItem item = area.solve(tempArr[i].getX(), tempArr[i].getY(), currentR);
            if (item!=null)
            {
                history.addItem(item);
            }
        }
    }

    //pointCount++;
}
```

```
}
```
HistoryBean.java
```
 package Beans;

import javax.annotation.PostConstruct;
import javax.faces.bean.ApplicationScoped;
import javax.faces.bean.ManagedBean;

import java.io.Serializable;
import java.util.Vector;


@ManagedBean(name="HistoryBean")
@ApplicationScoped
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();
   }
}
```
HistoryItem.java
```
 package Beans;

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;
    }
}
```

```
 package Beans;

import javax.faces.bean.ApplicationScoped;
import javax.faces.bean.ManagedBean;
import java.io.Serializable;

@ManagedBean(name="NavigationBean")
@ApplicationScoped
public class NavigationBean implements Serializable
{
    public String index()
    {
        return "index";
    }
    public String main()
    {
        return "main";
    }
}
```

ViewBean.java

```
 package Beans;

import Tools.ImgGenerator;
//import com.sun.msv.reader.Controller;
```

```
import org.primefaces.model.DefaultStreamedContent;
import org.primefaces.model.StreamedContent;
import javax.annotation.PostConstruct;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.ManagedProperty;
import javax.faces.bean.ApplicationScoped;
import javax.imageio.ImageIO;
import javax.inject.Inject;
import java.awt.image.BufferedImage;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.Serializable;

@ManagedBean(name="ViewBean")
@ApplicationScoped
public class ViewBean implements Serializable
{
   @ManagedProperty(value="#{HistoryBean}")
   private HistoryBean historyBean;

   private StreamedContent image;

   @ManagedProperty(value="#{ControllerBean}")
   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 = "IN";


   private static final String outAreaMSG = "OUT";
   private static final String TableFirstSTR ="<td>X</td> <td>Y</td> <td> R </td> <td>Result</td>";

   @PostConstruct
   public void init()
   {
      imageWidht  = 400;
      imageHeight = 400;
   }

   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());

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

            result = new DefaultStreamedContent(new ByteArrayInputStream(os.toByteArray()),
"image/png");
        }
        catch (IOException ex)
        {
            ex.printStackTrace();
        }

        return result;
    }

    String genHTMLTable()
    {
        StringBuilder build = new StringBuilder();
        build.append("<table>");
        build.append("<tr>" + TableFirstSTR + "</tr>");
        for(HistoryItem item : historyBean.toArray())
```

```
        {
            build.append("<tr>");
            build.append("<td>" + item.getX() + "</td><td>" + item.getY() + "</td> <td>" + item.getR() +
"</td>");
            build.append("<td>" + (item.getResult()?inAreaMSG:outAreaMSG) +"</td>");
            build.append("</tr>");
        }
        build.append("</table>");
        return build.toString();
    }
}
```

ImgGenerator.java

```
 package Tools;

import Beans.HistoryItem;

import java.awt.*;
import java.awt.image.BufferedImage;

public class ImgGenerator
{
    private static final double ImageResolutionR = 8;
    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.GRAY);

    g.fillArc(xc - R, yc - R, 2 * R, 2 * R, -90, -90);
    g.fillRect(xc, 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)
{
    Polygon result = null;
    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;
        y_points[1] = height/2;

        x_points[2] = width/2;
        y_points[2] = height/2 + R;

        result = new Polygon(x_points,y_points,num);
        return result;
    }
}
```

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">
 <navigation-rule>
        <navigation-case>
            <from-outcome>index</from-outcome>
            <to-view-id>/templates/index.xhtml</to-view-id>
        </navigation-case>
        <navigation-case>
            <from-outcome>main</from-outcome>
            <to-view-id>/templates/main.xhtml</to-view-id>
        </navigation-case>
    </navigation-rule>
</faces-config>
```

web.xml

```
 <?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" 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-
app_3_1.xsd">
    <servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>*.xhtml</url-pattern>
    </servlet-mapping>
    <session-config>
```

```
        <session-timeout>
            30
        </session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>templates/index.xhtml</welcome-file>
    </welcome-file-list>
</web-app>
```

Вывод: в процессе выполнения работы были изучены базовые аспекты построения приложения с помощью технологии jsf.