Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Факультет Программной инженерии и компьютерной техники

Кафедра Информатики, прикладной математики

Дисциплина: Програмирование интернет-приложений

*Лабораторная работа №3*

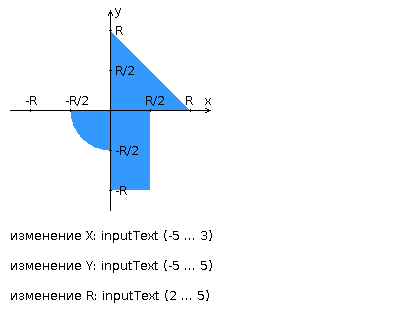
Группа №Р3217

Баев Д.В.

Преподаватель:

Николаев В.В.

Санкт-Петербург, 2017 г.



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 6 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](http://www.icesoft.org/java/projects/ICEfaces/overview.jsf" \t "_blank) (префикс "ace") и [PrimeFaces](http://www.primefaces.org/" \t "_blank) (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:**

* Все результаты проверки должны сохраняться в базе данных под управлением СУБД Oracle.
* Для доступа к БД необходимо использовать ORM Hibernate.
* Для управления списком 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.

**Основная страница:**





**faces-config:**

****

**hibernate.cfg:**

****

**CheckArea:**

|  |
| --- |
| public class CheckArea implements  java.io.Serializable { |
|  | Double x; |
|  | Double y; |
|  | Double r; |
|  | DataBase dataBase; |
|  | public CheckArea(){ |
|  | dataBase = new DataBase(); |
|  | } |
|  | public void setX(Double x){ |
|  | this.x=x; |
|  | } |
|  | public void setY(Double y){ |
|  | this.y=y; |
|  | } |
|  | public void setR(Double r){ |
|  | this.r=r; |
|  | } |
|  | public Double getX(){ |
|  | return x; |
|  | } |
|  | public Double getY(){ |
|  | return y; |
|  | } |
|  | public Double getR(){ |
|  | return r; |
|  | } |
|  | public String check() |
|  | { |
|  | if((y<=-x+r&&y>=0&&x>=0)||(y<=0&&y>=-r&&x>=0&&x<=r/2)||(x\*x+y\*y<=(r/2)\*(r/2)&&x<=0&&y<=0)) |
|  | return "in the area"; |
|  | return "not in the area"; |
|  | } |
|  | public String addToDb(){ |
|  | String res = check(); |
|  | dataBase.add(x,y,r,res); |
|  | return "add"; |
|  | } |
|  | public List getResponse(){ |
|  | List response; |
|  | response = dataBase.getAll(); |
|  | return response; |
|  | } |
|  | public void clear(){ |
|  | dataBase.clear(); |
|  | } |
|  |  |
|  | } |

**DataBase:**

|  |
| --- |
| public class DataBase implements java.io.Serializable { |
|  | public DataBase(){} |
|  |  |
|  | public void add(Double x,Double y,Double r,String result) { |
|  | Session session = null; |
|  | Transaction tx=null; |
|  | try { |
|  | session = HibernateUtil.getSessionFactory().openSession(); |
|  | tx = session.beginTransaction(); |
|  | PointsEntity points = new PointsEntity(); |
|  | points.setX(x); |
|  | points.setY(y); |
|  | points.setR(r); |
|  | points.setResult(result); |
|  | session.flush(); |
|  | session.save(points); |
|  | tx.commit(); |
|  | } catch (HibernateException e) { |
|  | if (tx!=null) tx.rollback(); |
|  | e.printStackTrace(); |
|  | } finally { |
|  | if (session != null && session.isOpen()) { |
|  |  |
|  | session.close(); |
|  | } |
|  | } |
|  | } |
|  | public void clear(){ |
|  | Session session = null; |
|  | Transaction tx=null; |
|  | try { |
|  | session = HibernateUtil.getSessionFactory().openSession(); |
|  | tx=session.beginTransaction(); |
|  | Query query =session.createQuery("delete from PointsEntity p"); |
|  | query.executeUpdate(); |
|  | tx.commit(); |
|  | } catch (HibernateException e) { |
|  | if (tx!=null) tx.rollback(); |
|  | e.printStackTrace(); |
|  | } finally { |
|  | if (session != null && session.isOpen()) { |
|  |  |
|  | session.close(); |
|  | } |
|  | } |
|  | } |
|  | public List getAll() { |
|  | Session session = null; |
|  | List points = new ArrayList<PointsEntity>(); |
|  | try { |
|  | session = HibernateUtil.getSessionFactory().openSession(); |
|  | Query q = session.createQuery("from PointsEntity p order by p.id"); |
|  | points = q.list(); |
|  | } catch (HibernateException e) { |
|  | e.printStackTrace(); |
|  | } finally { |
|  | if (session != null && session.isOpen()) { |
|  | session.close(); |
|  | } |
|  | } |
|  | return points; |
|  | } |
|  | } |

**PointsEntity:**

|  |
| --- |
| @Entity |
|  | @Table(name = "POINTS", schema = "S223868", catalog = "") |
|  | public class PointsEntity { |
|  | private long id; |
|  | private Double x; |
|  | private Double y; |
|  | private Double r; |
|  | private String result; |
|  | @Id |
|  | @GeneratedValue(strategy = GenerationType.SEQUENCE,generator = "points\_gen") |
|  | @SequenceGenerator(name = "points\_gen",sequenceName = "Points\_seq") |
|  | @Column(name = "ID") |
|  | public long getId() { |
|  | return id; |
|  | } |
|  |  |
|  | public void setId(long id) { |
|  | this.id = id; |
|  | } |
|  |  |
|  | @Basic |
|  | @Column(name = "X") |
|  | public Double getX() { |
|  | return x; |
|  | } |
|  |  |
|  | public void setX(Double x) { |
|  | this.x = x; |
|  | } |
|  |  |
|  | @Basic |
|  | @Column(name = "Y") |
|  | public Double getY() { |
|  | return y; |
|  | } |
|  |  |
|  | public void setY(Double y) { |
|  | this.y = y; |
|  | } |
|  |  |
|  | @Basic |
|  | @Column(name = "R") |
|  | public Double getR() { |
|  | return r; |
|  | } |
|  |  |
|  | public void setR(Double r) { |
|  | this.r = r; |
|  | } |
|  |  |
|  | @Basic |
|  | @Column(name = "RESULT") |
|  | public String getResult() { |
|  | return result; |
|  | } |
|  |  |
|  | public void setResult(String result) { |
|  | this.result = result; |
|  | } |

***Вывод:***

В ходе выполнения работы были изучены и применены на практике:

* фреймворк JSF, его особенности и структура.
* написание JavaBeans (Managed Beans) в рамках JSF.
* Технология ORM и сопутствующая библиотека Hibernate, для организации доступа к реляционным БД.