Факультет программной инженерии и компьютерной техники

Веб-программирование

Лабораторная работа №3

Вариант 13213

Преподаватель: Письмак Алексей Евгеньевич

Выполнил: Кульбако Артемий Юрьевич

Р3212

# 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.

**Стартовая страница должна содержать следующие элементы:**

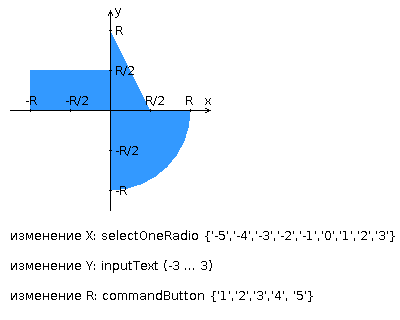
* "Шапку", содержащую ФИО студента, номер группы и номер варианта.
* Интерактивные часы, показывающие текущие дату и время, обновляющиеся раз в 10 секунд.
* Ссылку, позволяющую перейти на основную страницу приложения.

**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) (префикс "ace") и [PrimeFaces](http://www.primefaces.org) (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 PostgreSQL.
* Для доступа к БД необходимо использовать ORM EclipseLink.
* Для управления списком результатов должен использоваться Session-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.



# Выводы по работе.

В процессе работы я столкнулся с множеством проблем, большинство из которых связаны с странностями JSF.

* 1. Я не понимаю, как JSF можно считать хорошей реализацией MVC, если в разметке необходимо постоянно ссылаться на бины. В предыдущих заданиях разметка, логика фронтенда и бэкенда были абсолютно разделены: разметка содержала лишь элементы html, js осуществял ajax-запрос, сервер его обрабатывал и отправлял ответ. Каждый из участников был почти независим друг от друга. Подход jsf больше похож на процесс разработки классических, а не веб приложений, что непривычно и неудобно для веба.
  2. JSF меняет id всех элементов в форме на **имяФормы:вашId**. Проблема в том, что “**:**” - управляющий символ CSS. Если вы где-то в каскадных таблицах использовали id, придётся экранировать его вручную.

P.S. Позже я узнал, что это можно выключить, установив у формы свойство **prependId="false**", но какого чёрта это по-дефолту оно включено, неудобно же?!

* 1. Элементы из сторонних библиотек имею сложную структуру, из-за чего их сложно стилизовать. Пример: **p:selectOneRadio** представляет из себя миллион не **input type="radio"**, а миллион обёрнутых друг в друга div-ов.

Положительные эмоции вызвал JPA. Использовать его намного удобнее, чем вручную работать с БД.