Задача по технологиям J2EE

Краткое описание задачи

Задача состоит в написании на Java 2 EE Web-приложения – примитивной информационной системы. Индивидуальные варианты задачи отличаются тематикой информационной системы (набором таблиц в БД). Для выполнения задачи необходимы следующие знания: язык Java (J2SE: синтаксис, основные классы пакетов java.lang, java.util) , SQL (DML, простейшие запросы типа select, оператор create table), JDBC (создание соединения; выполнение SQL-инструкций; обработка наборов результатов), Java Servlets/Java Server Pages, Enterprise Java Beans (session beans и entity beans с Container Managed Persistence или Bean Managed Persistence), XML (опционально).

Технические средства

Следующие технические средства рекомендованы к реализации задачи.

СУБД: MySQL, Oracle

Сервер приложений/контейнер сервлетов/веб сервер: GlassFish, JBoss.

IDE: Eclipse, IntelliJ IDEA, NetBeans

Рекомендуется использовать следующие версии программного обеспечения jdk от 1.5 и выше , JBoss от 5.1 и выше, MySQL от 5.0, Oracle – XE. При использовании других версий и программ необходимо при сдаче задания иметь при себе дистрибутивы и самостоятельно производить установку, настройку и конфигурацию.

Структура базы данных

База данных состоит из двух таблиц, одна из которых (1) связана с другой (2) отношением «многие к одному» (имеет атрибут–ссылку). Таблица 2 является нередактируемой, а записи в первую (1) можно обновлять, удалять и связывать их с записями из 2 (связь является необязательной: nulls allowed). Все таблицы имеют собственный целочисленный первичный ключ, а также, по крайней мере, один неключевой строковый атрибут (имя/название). Таблицы предварительно создаются и заполняются через какую-либо внешнюю программу (например, через SQL Plus в случае Oracle).

Функции программы

1. Иметь заголовочную страницу с возможностью перехода на другие страницы приложения.
2. Выдавать полный список объектов таблицы 1 на странице, которая должна иметь возможность перехода на другие страницы приложения. Список выдавать в виде таблицы, содержащей все атрибуты объектов, а также (опционально) столбец с картинкой, при щелчке на который строка удаляется), исок заполняется заново).
3. Форма поиска. Формировать критерий поиска объектов таблицы 1: по части их имени либо по имени объекта таблицы 2, привязанного к ним. При этом пользователем задается любая подстрока имени (или первые символы строки

имени, как угодно). Выдавать список объектов (см. п.2) по нажатию кнопки из формы поиска.

4. Форма редактирования. Редактировать объекты 1, включая выбор связанного объекта 2 (из перечня имен таблицы 2) и допуская выбор отсутствующей связи (null). Переходить к редактированию новых объектов по отдельной ссылке со страниц 1 и 2, существующих объектов – по автоматически сгенерированной ссылке в строке таблицы вывода (см. п.2). Если объект нельзя удалить непосредственно из списка (таблицы), то должна быть возможность его удаления из формы редактирования.

Технические требования

1. Вся бизнес-логика по поиску объектов в БД по критериям, получению состояния (списка атрибутов) объектов, изменению состояния объектов и т.д. должна быть помещена в отдельный сервисный слой с декларативной поддержкой транзакций(stateless EJB или Spring bean).
2. Сервисный сслой не должен содержать запросов, осуществляющих поиск/создание/обновление (select, insert, update ...) объектов в БД напрямую. Необходимо создать слой хранения на любом ORM framework(например JPA или MyBatis) или чисто на JDBC для таблицы 1 (то же самое необходимо сделать и для таблицы 2) .
3. Слой представления (JSP, сервлеты, java script) должны обращаться только к сервисному слою и не посылать запросов напрямую в базу. Предлагается выбрать один из двух способов взаимодействия JSP и SessionBean
    - При запросе на список объектов или состояния одного объекта сервисный слой возвращает JSON или XML фиксированной схемы, которая разбирается в JSP странице для формирования HTML.
    - Вместо XML можно использовать простые классы(POJO) для хранения состояния объектов хранящихся в таблице 1, с полями, соответствующими столбцам таблицы 1 (то же самое сделать и для таблицы 2). В этом случае Session Bean возвращает один экземпляр или коллекцию экземпляров этого класса. Класс при этом не должен содержать никакой логики (получать доступ в базу и т.д.), он служить только для представления данных
4. Полный список и список объектов, отобранных по критерию, должны показываться с помощью одной и той же JSP-страницы.
5. Обработку запросов JSP-страниц желательно «поручить» нескольким сервлетам. Наличие одного метода-обработчика (типа doPost) размером в несколько экранов не приветствуется. Альтернативный вариант – JSP имеет отдельные секции для обработки запросов на чтение и запросов на изменение.
6. После обработки действия (создание объекта, удаление, изменение свойств) должно произойти перенаправление на страницу, находящуюся в режиме просмотра. То

есть нажатие Refresh в браузере на любой странице системы не должно приводить к повторному созданию/удалению/обновлению.

7. Не следует предполагать уникальность имен в таблице 1: вместо имен необходимо запоминать идентификаторы (невидимые ключевые атрибуты) объектов.
8. При добавлении строки таблицы (целочисленное) значение следующего идентификатора (первичного ключа таблицы) должно автоматически увеличиваться на 1: пользователь не должен вводить значения идентификаторов «вручную». Рекомендуется использовать для этого специфичные для СУБД средства (объект SEQUENCE в Oracle или функцию LAST_INSERT_ID() в MySQL).
9. Поддержка корректной кодировки символов в атрибутах объектов необязательна.

Требования к представлению системы

Для сдачи задание должно быть предоставлено следующее:

1. Предварительная конфигурация и демонстрация работы web-приложения, размещенного на сервере приложений (размещение должно производиться не средствами, интегрированными в IDE, а средствами сервера приложений). В идеале нужно иметь виртуальную машину с установленными средой и приложением
2. Исходный код программы. Методы должны обязательно откомментированы. Приветствуются комментарии совместимые с JavaDoc
3. Описание реализации – классы, методы, структура базы и т.д. В качестве дополнения приветствуется документ в формате JavaDoc

Варианты структуры БД (можно также взять любой свой вариант)

Атрибуты, которые обязательно использовать в программе, помечаются символом *.

Порядок таблиц здесь соответствует их порядку (1,2) в постановке задачи.

1. Институт (группы): Студент (ФИО*, группа*, тип стипендии, дата зачисления), Группа (Номер*, Факультет).
2. Библиотека (книгоучет): Экземпляр книги (инвентарный номер*, книга*), Книга (авторы, название*, раздел УДК, год издания, число страниц).
3. Отдел кадров: Сотрудник (ФИО*, отдел*, телефон (ы), зарплата), Отдел (название*, комната (ы), начальник).
4. Отдел кадров: Сотрудник (ФИО*, должность*, телефон (ы), зарплата), Должность (название*).
5. Отдел поставок: Сырье (название*, поставщик*, цена), Поставщик (название*, расчетный счет, ФИО контактного лица).
6. Отдел продаж: Заказ (номер*, заказчик*, дата, сумма заказа), Заказчик (Клиент) (название (или ФИО)*, телефон (ы)).
7. Ресторан: Блюдо (название*, категория*, цена), Категория блюд (название*).

8. Анализ публикаций: Публикация (название*, тип, аннотация, источник*, дата), Источник (название*, город, телефон(ы)).