Государственное образовательное учреждение
высшего профессионального образования

“ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПУТЕЙ СООБЩЕНИЯ”

Кафедра « Информационные вычислительные системы »

Курсовой проект по дисциплине «Базы данных»

Пояснительная записка

Разработал студент группы ПВТ - 711
Бабиев Д.П.

Руководитель: Хомоненко А.Д.

САНКТ-ПЕТЕРБУРГ
2010

Содержание

1. Описание предметной области.

В современном мире автомобиль является одним из самых надёжных и практичных видов передвижения. Существует множество компаний, которые занимаются их производством – General motors, Porsche, Renault, Nissan, Toyota, Ford и др. Зачастую продажей автомобилей занимаются не компании-производители, а их официальные дилеры.

Официальными или авторизованными дилерами принято называть, как правило, юридически самостоятельные предприятия, деятельность которых основана на продаже и обслуживании товаров одного или нескольких производителей.

Далеко не любая фирма может получить статус официального дилера. Компании должны обладать хорошей репутацией в данном районе рынка, достаточной материально-технической базой, опытом работы с аналогичными машинами. При подборе учитываются также их взаимоотношения с банками, организаторские способности и финансовые возможности для развития деловых операций, наличие квалифицированного персонала.

Обычно предусматривается, что дилеры будут наращивать продажи из года в год. Как правило, договорами предписываются квоты ежегодных или ежеквартальных продаж, которые должны обеспечить дилеры. Например, при заключении дилерского договора с компанией «Ford» четко оговариваются ежемесячные квоты на поставки автомобилей этой марки, причем базовой единицей отсчета в данном случае является количество всех проданных моделей за исключением популярной модели «Focus». Чем больше других моделей продал дилер, тем больше автомобилей «Ford Focus» будет отгружено производителем в новом месяце.

Выбор данной предметной области в курсовом проекте обуславливается тем, что для любого дилера важна оперативность в получении полной информации о продажах автомобилей во всех его офисах, которая отражает как финансовую, так и договорную сторону компании по поставкам тех или иных автомобилей.

В эту предметную область входят такие сущности как: офисы, менеджеры, производители, автомобили и др.

2. Выбор и описание средств разработки.

2.1. СУБД.

В курсовом проекте в качестве системы управления базой данных был выбран продукт компании Oracle – Oracle Database 10g XE.

Oracle Database 10g - первая в мире база данных, разработанная специально для работы в сетях распределенных вычислений. Oracle Database 10g предназначена для эффективного развертывания на базе различных типов оборудования, от небольших серверов до Oracle Enterprise Grid мощных многопроцессорных серверных систем, от отдельных кластеров до корпоративных распределенных вычислительных систем.

Oracle Database 10g позволяет пользователям виртуализировать использование аппаратного обеспечения - серверов и систем хранения данных. Oracle Database 10g обладает технологиями, которые позволяют администраторам надежно хранить и быстро распределять и извлекать данные для пользователей и приложений, работающих в сетях.

Oracle Database 10g предоставляет возможность автоматической настройки и управления, которая делает ее использование простым и экономически выгодным. Ее уникальные возможности осуществлять управление всеми данными предприятия - от обычных операций с бизнес-информацией до динамического многомерного анализа данных (OLAP), операций с документами формата XML, управления распределенной/локальной информацией - делает ее идеальным выбором для выполнения приложений, обеспечивающих обработку оперативных транзакций, интеллектуальный анализ информации, хранение данных и управление информационным наполнением.

СУБД Oracle Database 10*g* поставляется в четырех различных редакциях, ориентированных на различные сценарии разработки и развертывания приложений:

- ✓ Oracle Database 10*g* Standard Edition One
- ✓ Oracle Database 10*g* Standard Edition (SE)
- ✓ Oracle Database 10*g* Enterprise Edition (EE)
- ✓ Oracle Database 10*g* eXpress Edition (XE)

Кратко о редакции базы данных использованной в курсовом проекте.

Oracle Database XE - это СУБД начального уровня, основанная на базе кода Oracle Database 10g Release 2. Предоставляет возможность разработчикам ПО, администраторам баз данных и всем желающим получить бесплатную базовую версию СУБД, позволяющую начать разработку и развертывание собственных приложений. Кроме того, эта версия предлагается бесплатно независимым разработчикам ПО и поставщикам оборудования для свободной дистрибуции или встраивания в собственные приложения.

Oracle Database XE - это отличная СУБД начального уровня для:

- разработчиков, работающих с PHP, Java, .NET и Open Source-приложениями (альтернатива MySQL);

- администраторов баз данных, которым нужна бесплатная СУБД для обучения и инсталляций;
- независимых производителей ПО и аппаратных средств, которым нужна бесплатная СУБД для свободного распространения;
- образовательных учреждений и студентов, которым необходима бесплатная СУБД для обучения.

Oracle Database XE имеет следующие преимущества:

- абсолютно свободная СУБД;
- можно тестировать, разрабатывать и распространять с нулевыми инвестициями в ПО и без риска;
- легко мигрировать на промышленные редакции;
- не требуется переписывание приложений при переходе на промышленные редакции;

В данном курсовом проекте создаётся база данных, предоставляющая информацию о продажах различных автомобилей в сети офисов одного дилера. Пользователь разрабатываемого приложения (например, главный бухгалтер при составлении отчётов) сможет просматривать информацию об интересующих офисах в плане их продаж, такую как: дата продажи автомобиля; цена, за которую он был продан; менеджер, составивший договор продажи. В данной базе хранятся сведения о характеристиках продаваемых автомобилей: марка, модель, мощность, объём топливного бака, тип коробки передач и др. Есть возможность просмотреть всех производителей, с которыми компания-дилер сотрудничает.

2.2. Сервер приложений.

В курсовом проекте в качестве сервера приложений был выбран продукт компании Sun Microsystems – GlassFish v3.

GlassFish v3 - первый сертифицированный сервер приложений, поддерживающий стандарт Java EE 6, включающий в себя:

- enterprise-технологии: EJB 3.1, JPA 2.0, JDBC 4.0, CORBA 3.0;
- Web-технологии: Servlet 3.0, JSP 2.2, JSTL 1.2, EL 2.2, JSF 2.0 (Facelets), RESTful web services;

2.3. Среда разработки.

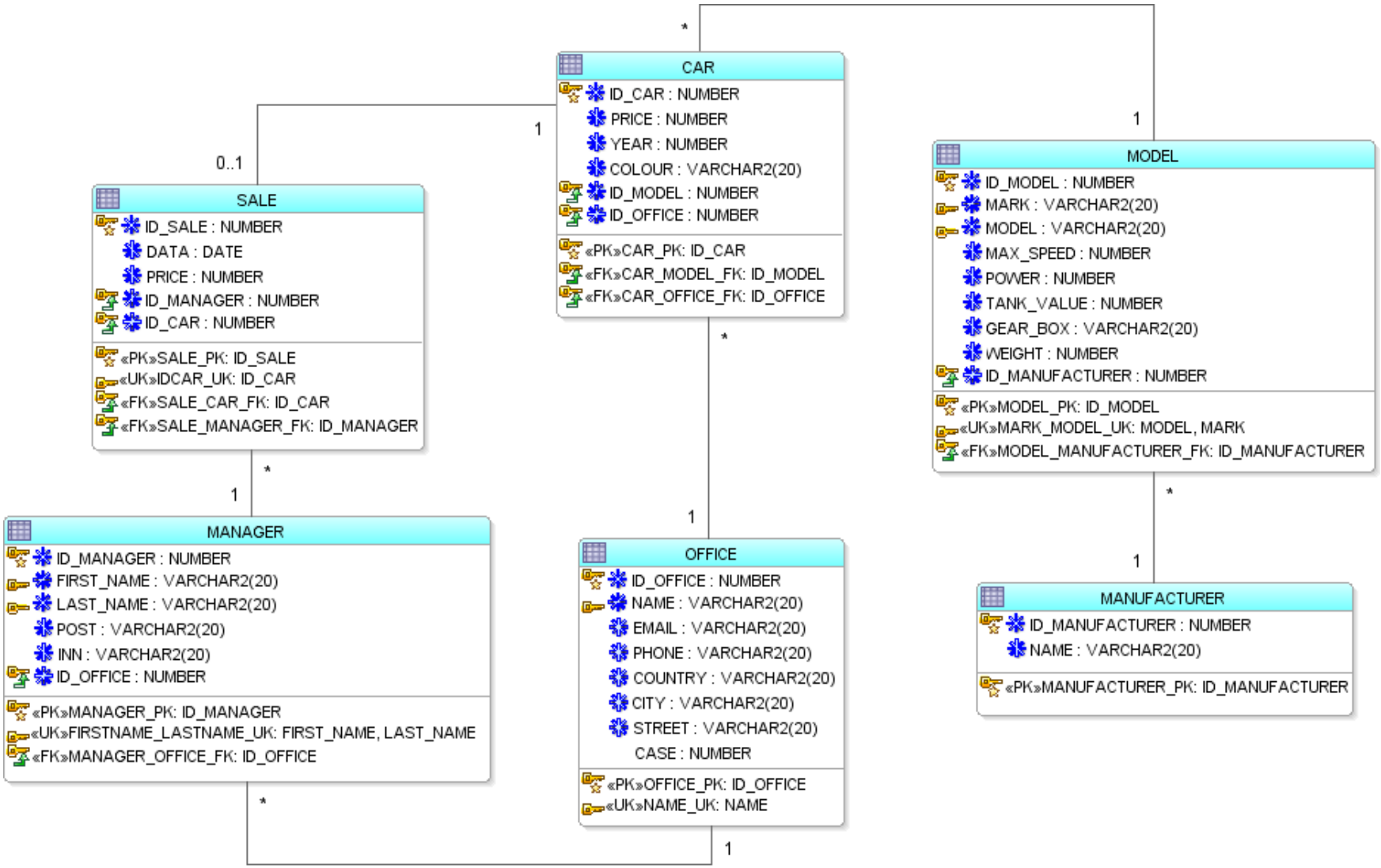
В курсовом проекте в качестве среды разработки был выбран продукт компании NetBeans Community – NetBeans 6.9.1.

NetBeans IDE — свободная интегрированная среда разработки приложений (IDE) на языках программирования Java, JavaFX, Ruby, Python, PHP, JavaScript, C++, Ада и ряде других. Данная среда поддерживает разработку Java Web - приложений, создание jsp, HTML страниц, XML документов, работу с базами данных.

Проект NetBeans IDE поддерживается и спонсируется компанией Oracle, однако разработка NetBeans ведется независимо сообществом разработчиков-энтузиастов (NetBeans Community) и компанией NetBeans Org.

3. Описание схемы БД. Реализация БД.

3.1. Схема базы данных.



3.2. Описание схемы базы данных.

3.1.1. Таблица MANUFACTURER

Таблица **Manufacturer** – содержит информацию о производителях автомобилей.

Атрибуты таблицы:

- ✓ ID_MANUFACTURER – идентификационный номер производителя.
- ✓ NAME – название производителя.

Ограничения таблицы:

- MANUFACTURER_PK: ID_MANUFACTURER – первичный ключ таблицы.

3.1.2. Таблица MODEL

Таблица **Model** – содержит подробную информацию о характеристиках моделей автомобилей.

Атрибуты таблицы:

- ✓ ID_MODEL – идентификационный номер модели.
- ✓ MARK – марка автомобиля.
- ✓ MODEL – модель автомобиля.
- ✓ MAX_SPEED – максимальная скорость автомобиля.
- ✓ POWER – мощность автомобиля.
- ✓ TANK_VALUE – объём топливного бака автомобиля.
- ✓ GEAR_BOX – коробка передач автомобиля.
- ✓ WEIGHT – масса автомобиля.
- ✓ ID_MANUFACTURER – ссылка на производителя.

Ограничения таблицы:

- MODEL_PK: ID_MODEL – первичный ключ таблицы.
- MARK_MODEL_UK: MODEL, MARK – уникальность комбинации Марка – Модель.
- MODEL_MANUFACTURER_FK: ID_MANUFACTURER – внешний ключ на **Manufacturer**

3.1.3. Таблица OFFICE

Таблица **Office** – содержит основную информацию об офисах.

Атрибуты таблицы:

- ✓ ID_OFFICE – идентификационный номер офиса.
- ✓ NAME – наименование офиса.
- ✓ EMAIL – электронный адрес офиса.
- ✓ PHONE – номер телефона офиса.
- ✓ COUNTRY – страна, где расположен офиса.
- ✓ CITY – город, где расположен офис.

- ✓ STREET – улица, где расположен офис.
- ✓ CASE – номер корпуса, где расположен офис.

Ограничения таблицы:

- OFFICE_PK: ID_OFFICE – первичный ключ таблицы.
- NAME_UK: NAME – уникальность наименования офиса.

3.1.4. Таблица MANAGER

Таблица **Manager** – содержит основную информацию о менеджерах.

Атрибуты таблицы:

- ✓ ID_MANAGER – идентификационный номер менеджера.
- ✓ FIRST_NAME – имя менеджера.
- ✓ LAST_NAME – фамилия менеджера.
- ✓ POST – пост менеджера.
- ✓ INN – ИНН менеджера.
- ✓ ID_OFFICE – ссылка на офис, где работает менеджер.

Ограничения таблицы:

- MANAGER_PK: ID_MANAGER – первичный ключ таблицы.
- FIRSTNAME_LASTNAME_UK: FIRST_NAME, LAST_NAME – уникальность комбинации Имя - Фамилия.
- MANAGER_OFFICE_FK: ID_OFFICE – внешний ключ на **Office**.

3.1.5. Таблица CAR

Таблица **Car** – содержит информацию об автомобилях офиса.

Атрибуты таблицы:

- ✓ ID_CAR – идентификационный номер автомобиля.
- ✓ PRICE – цена на автомобиль, заявленная офисом.
- ✓ YEAR – год производства автомобиля.
- ✓ COLOUR – цвет автомобиля.
- ✓ ID_MODEL – ссылка на модель автомобиля.
- ✓ ID_OFFICE – ссылка на офис, где продаётся автомобиль.

Ограничения таблицы:

- CAR_PK: ID_CAR – первичный ключ таблицы.
- CAR_MODEL_FK: ID_MODEL – внешний ключ на **Model**.
- CAR_OFFICE_FK: ID_OFFICE – внешний ключ на **Office**.

3.1.5. Таблица SALE

Таблица **Sale** – содержит информацию о продажах автомобилей.

Атрибуты таблицы:

- ✓ ID_SALE – идентификационный номер продажи.
- ✓ DATA – дата продажи.
- ✓ PRICE – цена, за которую был продан автомобиль.
- ✓ ID_MANAGER – ссылка на менеджера, продавшего автомобиль.
- ✓ ID_CAR – ссылка на автомобиль, который был продан.

Ограничения таблицы:

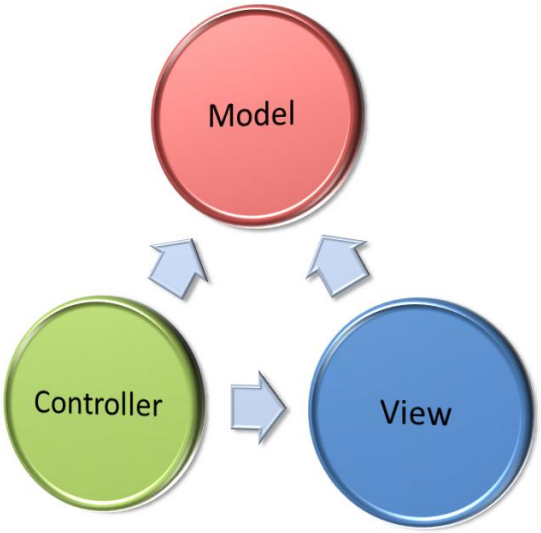
- SALE_PK: ID_SALE – первичный ключ таблицы.
- IDCAR_UK: ID_CAR – уникальность экземпляра продаваемого автомобиля.
- SALE_CAR_FK: ID_CAR – внешний ключ на **Car**.
- SALE_MANAGER_FK: ID_MANAGER – внешний ключ на **Manager**.

4. Разработка приложения.

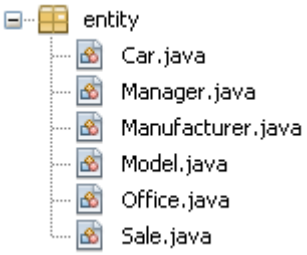
Разработка web-приложения базируется на концепции **MVC** (Model-View-Controller).

Model-view-controller (MVC, «Модель-представление-поведение», «Модель-представление-контроллер») — архитектура программного обеспечения, в которой модель данных приложения, пользовательский интерфейс и управляющая логика разделены на три отдельных компонента, так, что модификация одного из компонентов оказывает минимальное воздействие на другие компоненты. Шаблон *MVC* позволяет разделить данные, представление и обработку действий пользователя на три отдельных компонента

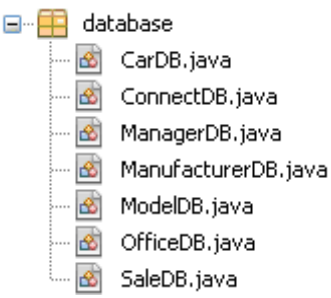
• Модель (**Model**). Модель предоставляет данные (обычно для View), а также реагирует на запросы (обычно от контроллера), изменяя своё состояние.
• Представление (**View**). Отвечает за отображение информации (пользовательский интерфейс).
• Поведение (**Controller**). Интерпретирует данные, введённые пользователем, и информирует модель и представление о необходимости соответствующей реакции.



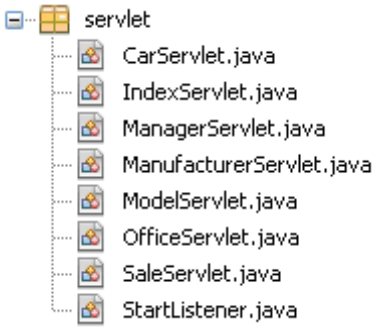
В соответствии с данной моделью были созданы 3 пакета: **entity**, **database**, **servlet**.



✓ **entity** - классы, представляющие собой модели таблиц БД (часть Model).



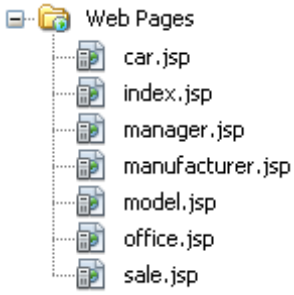
✓ **database** - классы, отвечающие за взаимодействие с БД (часть Model).



- ✓ **servlet** – классы-сервлеты (Controller), получающие данные, пришедшие на сервер после отправки формы из браузера и взаимодействующие с классами пакетов database и entity.

Servlet – является Java-программой, выполняющейся на стороне сервера и расширяющей функциональные возможности сервера. **Servlet** взаимодействует с клиентами посредством принципа запрос-ответ.

Роль View в трёхзвенной архитектуре занимают **jsp**-страницы.



JSP (**JavaServer Pages**) — технология, создания страниц, которые имеют как статические, так и динамические компоненты. Страница **JSP** является текстовым документом, который содержит:

- статическую часть (в формате HTML,XML).
- динамическую часть (использование Java-кода).

JSP — одна из высокопроизводительных технологий, так как весь код страницы транслируется в java-код сервлета, и затем компилируется в байт-код виртуальной машины java (JVM).

Для поддержания связи в проекте существует дескриптор развертывания, в виде XML-документа, в котором указывается адрес, при обращении к которому будет вызываться тот или иной сервлет. При вызове сервлет выполняет порученные ему функции и направляет результат на jsp страницу, которую просматривает пользователь.

5. Описание пользовательского интерфейса.

5.1. Главная страница.

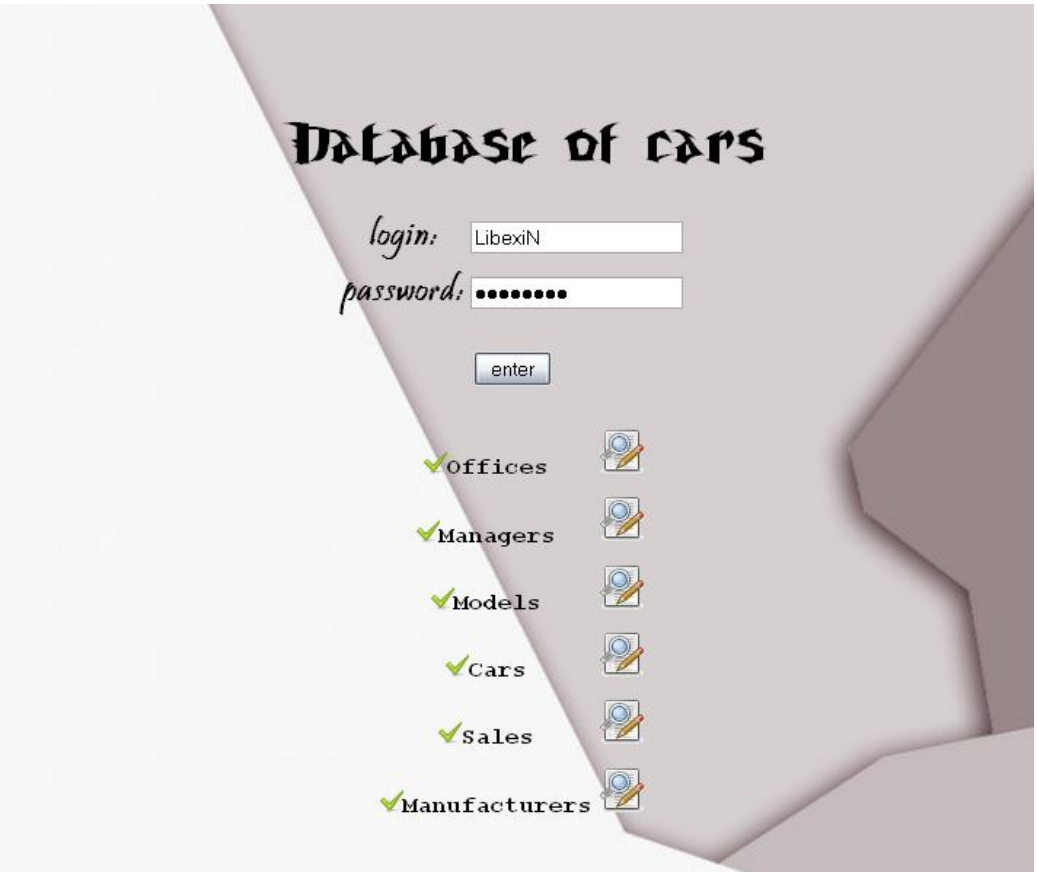


Рис. 5.1.1. Главная страница.

Назначение страницы:

- ✓ авторизация пользователя.
- ✓ выбор таблицы для просмотра/редактирования 

5.2. Страница просмотра/редактирования (на примере таблицы Manager).

Рис. 5.2.1. Страница просмотра/редактирования записей таблицы Manager.

Возможности на странице:

- ✓ просмотр всех записей таблицы Manager.
- ✓ переход на добавления нового менеджера
- ✓ переход к поиску менеджера
- ✓ переход к просмотру информации о продажах менеджера
- ✓ переход к редактированию менеджера
- ✓ переход к удалению менеджера
- ✓ сохранение изменений
- ✓ обновление страницы
- ✓ возврат на главную страницу

Рис. 5.2.2. Добавления нового менеджера.

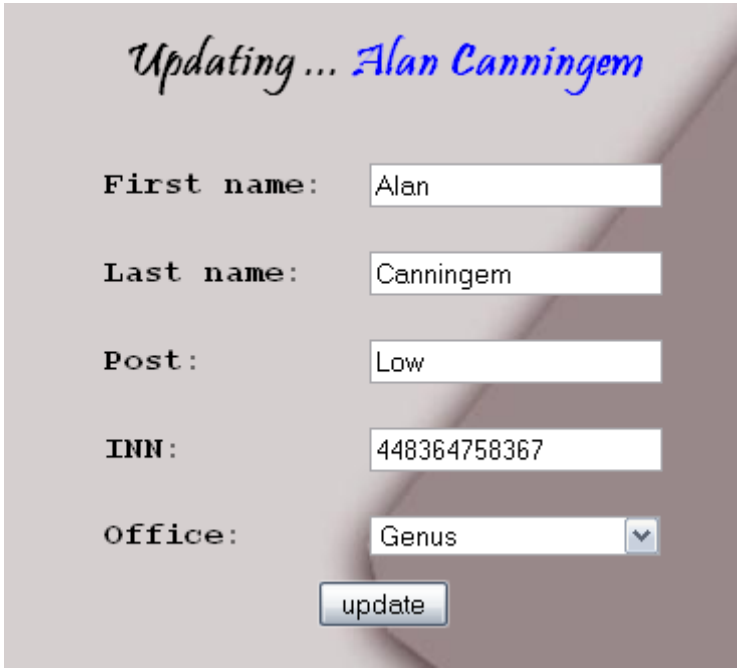


Рис. 5.2.3. Редактирования менеджера.

Рис. 5.2.4. Удаление менеджера.

Рис. 5.2.5. Поиск менеджера.

Рис. 5.2.6. Информация о продажах менеджера.

Список литературы

1. Java 2 SDK, Standard Edition Documentation, version 1.6.
2. Bruce W. Perry Java Servlet and JSP Cookbook. – M., 2006.
3. Лепехин А.Ф. Программирование Web-приложений. – СПб., 2002.

Приложение

1. Java - классы

1.1. Класс - прослушиватель изменений жизненного цикла сервлета.

```
package servlet;

import entity.*;
import database.*;

public class StartListener implements ServletContextListener
{
    @Override
    public void contextInitialized(ServletContextEvent sce)
    {
        ConnectDB connectDB = new ConnectDB();
        connectDB.createConnection(new User("LibexiN","dfktycbz"));

        OfficeDB officeDB = new OfficeDB(connectDB);
        ManagerDB managerDB = new ManagerDB(connectDB);
        CarDB carDB = new CarDB(connectDB);
        ModelDB modelDB = new ModelDB(connectDB);
        SaleDB saleDB = new SaleDB(connectDB);
        ManufacturerDB manufacturerDB = new ManufacturerDB(connectDB);

        sce.getServletContext().setAttribute("connect", connectDB);
        sce.getServletContext().setAttribute("office", officeDB);
        sce.getServletContext().setAttribute("manager", managerDB);
        sce.getServletContext().setAttribute("car", carDB);
        sce.getServletContext().setAttribute("model", modelDB);
        sce.getServletContext().setAttribute("sale", saleDB);
        sce.getServletContext().setAttribute("manufacturer", manufacturerDB);
    }

    @Override
    public void contextDestroyed(ServletContextEvent sce)
    {
        sce.getServletContext().removeAttribute("office");
        sce.getServletContext().removeAttribute("manager");
        sce.getServletContext().removeAttribute("car");
        sce.getServletContext().removeAttribute("model");
        sce.getServletContext().removeAttribute("sale");
    }
}
```

1.2. Класс подключения к базе данных Oracle.

```
package database;

public class ConnectDB
{
    private Connection myConnect = null;
    private Statement myStmt = null;
    private DatabaseMetaData myMetaData = null;
    private User user = null;

    public ConnectDB(){}

    public boolean createConnection(User user)
    {
        this.user = user;
        try
        {
            if(myConnect != null)
            {
                myConnect.close();
                myConnect = null;
            }
            Class.forName("oracle.jdbc.driver.OracleDriver");
            myConnect = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE",
                                                    this.user.getLogin(),
                                                    this.user.getPassword());
            if(myConnect == null)
                return false;
            myConnect.setAutoCommit(false);
            myStmt = myConnect.createStatement();
            if(myStmt == null)
```

```
                return false;
            return true;
        }
        catch (Exception e)
        {
            System.err.println("JDBC connecting: ");
            System.err.println(e);
            return false;
        }
    }

    public Connection getConnect() {return myConnect;}
    public Statement getStmt() {return myStmt;}
    public User getUser(){return user;}

    public boolean isConnection()
    {
        try{return !myConnect.isClosed();}
        catch (Exception e) {return false;}
    }

    public void commit()
    {
        try{myConnect.commit();}
        catch (Exception e) {}
    }

    public void rollback()
    {
        try{myConnect.rollback();}
        catch (Exception e) {}
    }

    public void destroyConnection()
    {
        Try {myConnect.close();}
        catch (Exception e) {}
    }
}
```

1.3. Класс сущности Manager.

```
package entity;

public class Manager
{
    private int ID_MANAGER;
    private String FIRST_NAME;
    private String LAST_NAME;
    private String POST;
    private String INN;
    private int ID_OFFICE;

    public Manager() {}

    public Manager(int ID_MANAGER, String FIRST_NAME, String LAST_NAME,
            String POST, String INN, int ID_OFFICE)
    {
        this.ID_MANAGER = ID_MANAGER;
        this.FIRST_NAME = FIRST_NAME;
        this.LAST_NAME = LAST_NAME;
        this.POST = POST;
        this.INN = INN;
        this.ID_OFFICE = ID_OFFICE;
    }

    public void setID_MANAGER(int ID_MANAGER) {this.ID_MANAGER = ID_MANAGER;}
    public void setFIRST_NAME(String FIRST_NAME) {this.FIRST_NAME = FIRST_NAME;}
    public void setLAST_NAME(String LAST_NAME) {this.LAST_NAME = LAST_NAME;}
    public void setPOST(String POST) {.POST = POST;}
    public void setINN(String INN) {this.INN = INN;}
    public void setID_OFFICE(int ID_OFFICE) {this.ID_OFFICE = ID_OFFICE;}

    public int getID_MANAGER() {return ID_MANAGER;}
    public String getFIRST_NAME() {return FIRST_NAME;}
    public String getLAST_NAME() {return LAST_NAME;}
    public String getPOST() {return POST;}
    public String getINN() {return INN;}
    public int getID_OFFICE() {return ID_OFFICE;}
}
```

1.4. Класс запросов к базе данных для сущности Manager.

```
package database;

import entity.Manager;

public class ManagerDB
{
    private Connection myConnect = null;
    private Statement myStmt = null;

    public ManagerDB(ConnectDB myConnectDB)
    {
        myConnect = myConnectDB.getConnect();
        myStmt = myConnectDB.getStmt();
    }

    public ManagerDB() {}

    public List loadRecords(String ID_MANAGER, String ID_OFFICE) throws Exception
    {
        List<Manager> managers = new LinkedList<Manager>();
        if(!myConnect.isClosed())
        {
            ResultSet rs;
            if(ID_MANAGER != null)
                rs = myStmt.executeQuery("SELECT * FROM manager WHERE id_manager = " + ID_MANAGER);
            else if (ID_OFFICE != null)
                rs = myStmt.executeQuery("SELECT * FROM manager WHERE id_office = " + ID_OFFICE);
            else
                rs = myStmt.executeQuery("SELECT * FROM manager ORDER BY first_name,last_name");
            while(rs.next())
            {
                Manager manager = new Manager(rs.getInt("ID_MANAGER"),
                                              rs.getString("FIRST_NAME"),
                                              rs.getString("LAST_NAME"),
                                              rs.getString("POST"),
                                              rs.getString("INN"),
                                              rs.getInt("ID_OFFICE"));
                managers.add(manager);
            }
        }
        return managers;
    }

    public List searchRecords(String[] arrayOption)
    {
        try
        {
            List<Manager> managers = new LinkedList<Manager>();
            String query = "";
            if (arrayOption[0] !=  null)
                query += ("first_name=\'" + arrayOption[0] + "\' AND ");
            if(arrayOption[1] !=  null)
                query += ("last_name=" + " \'" + arrayOption[1] + "\' AND ");
            if(arrayOption[2] !=  null)
                query += ("post=" + " \'" + arrayOption[2] + "\' AND ");
            query = query.substring(0, (query.length()-5));
            ResultSet rs = myStmt.executeQuery
                          ("SELECT * FROM manager WHERE " + query + "ORDER BY first_name,last_name");
            while (rs.next())
            {
                Manager manager = new Manager(rs.getInt("ID_MANAGER"),
                                              rs.getString("FIRST_NAME"),
                                              rs.getString("LAST_NAME"),
                                              rs.getString("POST"),
                                              rs.getString("INN"),
                                              rs.getInt("ID_OFFICE"));
                managers.add(manager);
            }
            return managers;
        }
        catch (Exception ex) {return null;}
    }

    public void addManager(int ID_OFFICE, String FIRST_NAME, String LAST_NAME,
                   String POST, String INN) throws SQLException
    {
        try
        {
```

```
            if (!myConnect.isClosed())
            {
                PreparedStatement prst = myConnect.prepareStatement
                                    ("INSERT INTO manager VALUES (MANAGER_SEQ.NEXTVAL,?1,?2,?3,?4,?5)");
                prst.setString(1, FIRST_NAME);
                prst.setString(2, LAST_NAME);
                prst.setString(3, POST);
                prst.setString(4, INN);
                prst.setInt(5, ID_OFFICE);
                prst.execute();
                prst.close();
            }
        }
        catch (Exception ex) {return;}
    }

    public void updateManager(int ID_MANAGER, String FIRST_NAME, String LAST_NAME, String POST,
                  String INN, int ID_OFFICE) throws Exception
    {
        try
        {
            if (!myConnect.isClosed())
            {
                PreparedStatement prst = myConnect.prepareStatement
("UPDATE manager SET first_name=?1, last_name=?2, post=?3, inn=?4, id_office=?5 WHERE id_manager = ?6");
                prst.setString(1, FIRST_NAME);
                prst.setString(2, LAST_NAME);
                prst.setString(3, POST);
                prst.setString(4, INN);
                prst.setInt(5, ID_OFFICE);
                prst.setInt(6, ID_MANAGER);
                prst.execute();
                prst.close();
            }
        }
        catch (Exception ex) {return;}
    }

    public void deleteManager(int ID_MANAGER) throws Exception
    {
        if (!myConnect.isClosed())
        {
            PreparedStatement prst_sale =
                myConnect.prepareStatement("DELETE FROM sale WHERE id_manager = " + ID_MANAGER);
            PreparedStatement prst_manager =
                myConnect.prepareStatement("DELETE FROM manager WHERE id_manager = " + ID_MANAGER);
            prst_sale.execute();
            prst_manager.execute();
            prst_sale.close();
            prst_manager.close();
        }
    }
}
```

1.5. Класс для взаимодействия с браузером.

```
package servlet;

import database.*;

public class ManagerServlet extends HttpServlet
{
    ConnectDB connectDB;
    ManagerDB managerDB;
    OfficeDB officeDB;
    SaleDB saleDB;
    CarDB carDB;
    ModelDB modelDB;

    protected void processRequest(HttpServletRequest request, HttpServletResponse response) throws Exception
    {
        response.setContentType("text/html;charset=UTF-8");
        request.setCharacterEncoding("UTF-8");
        connectDB = (ConnectDB) this.getServletContext().getAttribute("connect");
        managerDB = (ManagerDB) this.getServletContext().getAttribute("manager");
        officeDB = (OfficeDB) this.getServletContext().getAttribute("office");
        saleDB = (SaleDB) this.getServletContext().getAttribute("sale");
        carDB = (CarDB) this.getServletContext().getAttribute("car");
        modelDB = (ModelDB) this.getServletContext().getAttribute("model");
```

```
    if(request.getParameter("view_page") != null)
    {
        String ID_MANAGER = request.getParameter("id_manager");
        request.setAttribute("selected_manager", managerDB.loadRecords(ID_MANAGER,null));
        request.setAttribute("manager_sales", saleDB.loadRecords(null,null,ID_MANAGER));
        request.setAttribute("cars_saled", carDB.loadRecords(null,ID_MANAGER,null,null));
        request.setAttribute("models_saled", modelDB.loadRecords(null,null,ID_MANAGER,null));
        request.setAttribute("offices", officeDB.loadRecords(null));
        request.setAttribute("managers", managerDB.loadRecords(null,null));
        request.setAttribute("view_page", "..."); // enable view page
    }
    else if(request.getParameter("edit_page") != null)
    {
        if(request.getParameter("add") != null)
        {
            if(request.getParameter("set_add") != null)
                procAdd(request);
            else
            {
                request.setAttribute("offices", officeDB.loadRecords(null));
                request.setAttribute("flagAdd", "...");
            }
        }
        else if(request.getParameter("update") != null)
        {
            if(request.getParameter("set_update") != null)
                procUpdate(request);
            else
            {
                String ID_MANAGER = request.getParameter("id_manager");
                request.setAttribute("selected_manager", managerDB.loadRecords(ID_MANAGER,null));
                request.setAttribute("flagUpdate", "...");
            }
        }
        else if(request.getParameter("delete") != null)
        {
            if(request.getParameter("delete_yes") != null)
                procDelete(request);
            else if(request.getParameter("delete_no") == null)
            {
                String ID_MANAGER = request.getParameter("id_manager");
                request.setAttribute("selected_manager", managerDB.loadRecords(ID_MANAGER,null));
                request.setAttribute("flagDelete", "...");
            }
        }
        else if(request.getParameter("commit") != null)
        {
            connectDB.commit();
        }
        request.setAttribute("offices", officeDB.loadRecords(null));
        request.setAttribute("managers", managerDB.loadRecords(null,null));
        request.setAttribute("edit_page", "..."); // enable edit page
    }
    else if(request.getParameter("search_page") != null)
    {
        if(request.getParameter("set_search") != null)
        {
            String arrayOption[] = {null,null,null}; // first_name,last_name,post
            if(request.getParameter("first_name_check") != null)
                arrayOption[0] = request.getParameter("first_name");
            if(request.getParameter("last_name_check") != null)
                arrayOption[1] = request.getParameter("last_name");
            if(request.getParameter("post_check") != null)
                arrayOption[2] = request.getParameter("post");
            request.setAttribute("offices", officeDB.loadRecords(null));
            request.setAttribute("find_managers", managerDB.searchRecords(arrayOption));
        }
        request.setAttribute("search_page", "..."); // enable search page
    }
    this.getServletContext().getRequestDispatcher("/manager.jsp").forward(request, response);
}

@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws Exception
{
    try{processRequest(request, response);}
    catch (Exception ex){Logger.getLogger(ManagerServlet.class.getName()).log(Level.SEVERE, null, ex);}
}

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws Exception
{
    try{processRequest(request, response);}
```

```
    catch (Exception ex){Logger.getLogger(ManagerServlet.class.getName()).log(Level.SEVERE, null, ex);}
}

public void procAdd(HttpServletRequest request) throws Exception
{
    String ID_OFFICE = request.getParameter("office");
    String FIRST_NAME = request.getParameter("first_name");
    String LAST_NAME = request.getParameter("last_name");
    String POST = request.getParameter("post");
    String INN = request.getParameter("inn");

    if(ID_OFFICE == null || ID_OFFICE.equals("_blank") || FIRST_NAME == null ||
       LAST_NAME == null || POST == null || INN == null)
        return;
    try{managerDB.addManager(Integer.parseInt(ID_OFFICE), FIRST_NAME, LAST_NAME, POST, INN);}
    catch(Exception ex){return;}
}

public void procUpdate(HttpServletRequest request) throws Exception
{
    String ID_MANAGER = request.getParameter("id_manager");
    String FIRST_NAME = request.getParameter("new_first_name");
    String LAST_NAME = request.getParameter("new_last_name");
    String POST = request.getParameter("new_post");
    String INN = request.getParameter("new_inn");
    String ID_OFFICE = request.getParameter("new_office");

    if (ID_OFFICE == null || ID_OFFICE.equals("_blank") || ID_MANAGER == null ||
        FIRST_NAME == null || LAST_NAME == null || POST == null || INN == null)
        return;
    try{managerDB.updateManager(Integer.parseInt(ID_MANAGER), FIRST_NAME, LAST_NAME,
                                POST, INN, Integer.parseInt(ID_OFFICE));}
    catch(Exception ex){return;}
}

public boolean procDelete(HttpServletRequest request) throws Exception
{
    String ID_MANAGER = (String) request.getParameter("id_manager");
    if (ID_MANAGER == null)
        return false;
    managerDB.deleteManager(Integer.parseInt(ID_MANAGER));
    return true;
}
```

1.6. JSP – страница для ответа браузеру.

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <LINK Rel="stylesheet" Type = "text/css" Href="./resource/style.css">
        <title>Page</title>
    </head>
    <body>
        <a href="IndexServlet?authorization_page"><img src="resource/picture/go-home.gif" width="32" height="32" alt="go-home"/></a>
        <c:if test="${requestScope.view_page != null}">
                <a href="ManagerServlet?edit_page"><img src="resource/picture/left_32.png" width="32" height="32" alt="left_32"/></a>
            <table align="center">
                <tr>
                    <td align="center">
                        <h2>Cars sold by manager:</h2>
                        <table class="StayleDataTable">
                            <tr bgcolor=yellow>
                                <th>Mark</th>
                                <th>Model</th>
                                <th>Color</th>
                                <th>Year</th>
                                <th>Data</th>
                            </tr>
                            <c:forEach items="${requestScope.manager_sales}" var="temp_sale">
                                <tr>
                                    <c:forEach items="${requestScope.cars_saled}" var="temp_cars_saled">
                                        <c:forEach items="${requestScope.models_saled}" var="temp_models_saled">
                                            <c:if test="${temp_models_saled.ID_MODEL == temp_cars_saled.ID_MODEL &&
                                                        temp_cars_saled.ID_CAR == temp_sale.ID_CAR}">
                                                <td>${temp_models_saled.MARK}</td>
                                                <td>${temp_models_saled.MODEL}</td>
                                            </c:if>
                                        </c:forEach>
                                        <c:if test="${temp_cars_saled.ID_CAR == temp_sale.ID_CAR}">
                                            <td>${temp_cars_saled.COLOUR}</td>
                                            <td>${temp_cars_saled.YEAR}</td>
                                        </c:if>
                                    </c:forEach>
                                    <td>${temp_sale.DATA}</td>
                                </tr>
                            </c:forEach>
                        </table><br>
                    </td>
                </tr>
            </table>
        </c:if>
        <c:if test="${requestScope.edit_page != null}">
            <a href="ManagerServlet?edit_page"><img src="resource/picture/view-refresh.gif" width="32" height="32" alt="view-refresh"/></a>
            <a href="ManagerServlet?edit_page&commit"><img src="resource/picture/save_32.png" width="32" height="32" alt="save_32"/></a>
            <table align="center">
                <tr>
```

```
<td valign="top" width="50%">
    <h2>All managers:</h2>
    <form action="" method="GET">
        <a href="<c:url value="/ManagerServlet"/>?edit_page&add">
            <img src="resource/picture/plus_32.png" width="32" height="32" alt="plus_32"/>
        </a>
        <a href="<c:url value="/ManagerServlet"/>?search_page">
            <img src="resource/picture/search_32.png" width="32" height="32" alt="search_32"/>
        </a>
        <table class="StayleDataTable">
            <tr>
                <th>First name</th>
                <th>Last name</th>
                <th>Post</th>
                <th>INN</th>
                <th>Office</th>
            </tr>
            <c:forEach items="${requestScope.managers}" var="temp_manager">
                <tr>
                    <td>${temp_manager.FIRST_NAME}</td>
                    <td>${temp_manager.LAST_NAME}</td>
                    <td>${temp_manager.POST}</td>
                    <td>${temp_manager.INN}</td>
                    <td>
                        <c:forEach items="${requestScope.offices}" var="temp_office">
                            <c:if test="${temp_office.ID_OFFICE == temp_manager.ID_OFFICE}">
                                ${temp_office.NAME}
                            </c:if>
                        </c:forEach>
                    </td>
                    <td>
                        <a href="<c:url value="/ManagerServlet"/>?view_page&id_manager=${temp_manager.ID_MANAGER}">
                            <img src="resource/picture/bubble_16.png" width="16" height="16" alt="bubble_16"/>
                        </a>
                        <a href="<c:url value="/ManagerServlet"/>?edit_page&id_manager=${temp_manager.ID_MANAGER}&update">
                            <img src="resource/picture/pencil_16.png" width="16" height="16" alt="pencil_16"/>
                       </a>
                       <a href="<c:url value="/ManagerServlet"/>?edit_page&id_manager=${temp_manager.ID_MANAGER}&delete">
                            <img src="resource/picture/delete_16.png" width="16" height="16" alt="delete_16"/>
                       </a>
                    </td>
                </tr>
            </c:forEach>
        </table>
    </form>
</td>
<td  valign="top" align="center" width="50%">
    <c:if test="${requestScope.flagAdd != null}">
        <h2>Add new manager:</h2>
        <h4>
            <form action="?edit_page&add&set_add" method="POST">
                <table cellpadding="10px">
                    <tr><td>First name:</td><td><input type="text" name="first_name" value="" /></td></tr>
                    <tr><td>Last name:</td><td><input type="text" name="last_name" value="" /></td></tr>
                    <tr><td>Post:</td><td><input type="text" name="post" value="" /></td></tr>
```

```
                <tr><td>INN:</td><td><input type="text" name="inn" value="" /></td></tr>
                <tr><td>Office:</td><td>
                        <select style="width:145px" name="office">
                            <option value="_blank">...</option>
                            <c:forEach items="${requestScope.offices}" var="temp_office">
                                <option value="${temp_office.ID_OFFICE}">
                                    ${temp_office.NAME}
                                </option>
                            </c:forEach>
                        </select></td></tr>
            </table>
            <input type="submit" value="add" name="but_add" />
        </form>
    </h4>
</c:if>
<c:if test="${requestScope.flagUpdate != null}">
    <form action="?edit_page&update&set_update" method="POST">
        <c:forEach items="${requestScope.selected_manager}" var="selected_manager">
            <h2>Updating ...
                <font color="blue"><b><c:out value="${selected_manager.FIRST_NAME}"></c:out>
                                    <c:out value="${selected_manager.LAST_NAME}"></c:out></b></font>
            </h2>
            <input type="hidden" name="id_manager" value="${selected_manager.ID_MANAGER}" />
            <h4>
                <table cellpadding="10px">
                    <tr><td>First name:</td><td><input type="text" name="new_first_name" value="${selected_manager.FIRST_NAME}"/></td></tr>
                    <tr><td>Last name:</td><td><input type="text" name="new_last_name" value="${selected_manager.LAST_NAME}"/></td></tr>
                    <tr><td>Post:</td><td><input type="text" name="new_post" value="${selected_manager.POST}" /></td></tr>
                    <tr><td>INN:</td><td><input type="text" name="new_inn" value="${selected_manager.INN}" /></td></tr>
                    <tr><td>Office:</td><td>
                            <select name="new_office">
                                <c:forEach items="${requestScope.offices}" var="temp_office">
                                    <c:if test="${selected_manager.ID_OFFICE == temp_office.ID_OFFICE}">
                                        <option selected value="${temp_office.ID_OFFICE}">
                                            ${temp_office.NAME}
                                        </option>
                                    </c:if>
                                    <c:if test="${selected_manager.ID_OFFICE != temp_office.ID_OFFICE}">
                                        <option value="${temp_office.ID_OFFICE}">
                                            ${temp_office.NAME}
                                        </option>
                                    </c:if>
                                </c:forEach>
                            </select>
                        </td>
                    </tr>
                </table>
            </h4>
            <input type="submit" value="update" name="but_update" />
        </c:forEach>
    </form>
</c:if>
<c:if test="${requestScope.flagDelete != null}">
    <c:forEach items="${requestScope.selected_manager}" var="selected_manager">
        <h2>
```

```
                        Deleting ... <font color="blue"><b><c:out value="${selected_manager.FIRST_NAME}"></c:out>
                                                        <c:out value="${selected_manager.LAST_NAME}"></c:out></b></font>
                    </h2>
                    <table>
                        <tr>
                            <td><form action="?edit_page&delete&delete_yes" method="POST">
                                    <input type="hidden" name="id_manager" value="${selected_manager.ID_MANAGER}" />
                                    <input type="submit" value="yes" />
                                </form></td>
                            <td><form action="?edit_page&delete&delete_no" method="POST">
                                    <input type="submit" value="no" />
                                </form></td>
                        </tr>
                    </table>
                </c:forEach>
            </c:if>
        </td>
    </tr>
</table>
</c:if>
<c:if test="${requestScope.search_page != null}">
    <a href="ManagerServlet?edit_page"><img src="resource/picture/left_32.png" width="32" height="32" alt="left_32"/></a>
    <table>
        <td valign="top" align="center" width="35%">
            <h2>Search options:</h2>
            <h4>
                <form action="?search_page&set_search" method="POST">
                    <table cellpadding="10px">
                        <tr>
                            <td><input type="checkbox" name="first_name_check" value="ON" /></td>
                            <td>First name:</td>
                            <td></td>
                            <td><input type="text" name="first_name" value="" /></td>
                        </tr>
                        <tr>
                            <td><input type="checkbox" name="last_name_check" value="ON" /></td>
                            <td>Last name:</td>
                            <td></td>
                            <td><input type="text" name="last_name" value="" /></td>
                        </tr>
                        <tr>
                            <td><input type="checkbox" name="post_check" value="ON" /></td>
                            <td>Post:</td>
                            <td></td>
                            <td>
                                <select style="width:145px" name="post">
                                    <option value="High">High</option>
                                    <option value="Low">Low</option>
                                </select>
                            </td>
                        </tr>
                    </table>
                    <input type="submit" value="search" name="but_search" />
                </form>
            </h4>
```

```
                </td>
                <td valign="top" align="center" width="35%">
                    <h2>Search result:</h2><br>
                    <table class="StayleDataTable">
                        <tr>
                            <th>First name</th>
                            <th>Last name</th>
                            <th>Post</th>
                            <th>Office</th>
                        </tr>
                        <c:forEach items="${requestScope.find_managers}" var="temp_manager">
                            <tr>
                                <td>${temp_manager.FIRST_NAME}</td>
                                <td>${temp_manager.LAST_NAME}</td>
                                <td>${temp_manager.POST}</td>
                                <td>
                                    <c:forEach items="${requestScope.offices}" var="temp_office">
                                        <c:if test="${temp_office.ID_OFFICE == temp_manager.ID_OFFICE}">
                                            ${temp_office.NAME}
                                        </c:if>
                                    </c:forEach>
                                </td>
                                <td>
                                    <a href="<c:url value="/ManagerServlet"/>?view_page&from_search_page&id_manager=${temp_manager.ID_MANAGER}">
                                       <img src="resource/picture/bubble_16.png" width="16" height="16" alt="bubble_16"/>
                                    </a>
                                </td>
                            </tr>
                        </c:forEach>
                    </table>
                </td>
            </table>
        </c:if>
    </body>
</html>
```

2. XML – документы.

2.1. Дескриптор развёртывания.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">
    <listener>
        <listener-class>servlet.StartListener</listener-class>
    </listener>
    <servlet>
        <servlet-name>OfficeServlet</servlet-name>
        <servlet-class>servlet.OfficeServlet</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet>
        <servlet-name>AuthorizationServlet</servlet-name>
        <servlet-class>servlet.AuthorizationServlet</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>UserServlet</servlet-name>
        <servlet-class>servlet.UserServlet</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>ManagerServlet</servlet-name>
        <servlet-class>servlet.ManagerServlet</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>ModelServlet</servlet-name>
        <servlet-class>servlet.ModelServlet</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>ManufacturerServlet</servlet-name>
        <servlet-class>servlet.ManufacturerServlet</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>IndexServlet</servlet-name>
        <servlet-class>servlet.IndexServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet>
        <servlet-name>CarServlet</servlet-name>
        <servlet-class>servlet.CarServlet</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>SaleServlet</servlet-name>
        <servlet-class>servlet.SaleServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>/faces/*</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>AuthorizationServlet</servlet-name>
        <url-pattern>/AuthorizationServlet</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>UserServlet</servlet-name>
        <url-pattern>/UserServlet</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>ManagerServlet</servlet-name>
        <url-pattern>/ManagerServlet</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>OfficeServlet</servlet-name>
        <url-pattern>/OfficeServlet</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>ModelServlet</servlet-name>
        <url-pattern>/ModelServlet</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>ManufacturerServlet</servlet-name>
        <url-pattern>/ManufacturerServlet</url-pattern>
```

```
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>IndexServlet</servlet-name>
        <url-pattern>/IndexServlet</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>CarServlet</servlet-name>
        <url-pattern>/CarServlet</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>SaleServlet</servlet-name>
        <url-pattern>/SaleServlet</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>faces/index.xhtml</welcome-file>
    </welcome-file-list>
</web-app>
```

3. SQL – запросы к базе данных.

3.1. Создание таблиц, ограничителей и последовательностей.

```
CREATE TABLE SALE
(
  ID_SALE NUMBER NOT NULL,
  DATA DATE NOT NULL,
  PRICE NUMBER NOT NULL,
  ID_MANAGER NUMBER NOT NULL,
  ID_CAR NUMBER NOT NULL
, CONSTRAINT SALE_PK PRIMARY KEY
  (
    ID_SALE
  )
  ENABLE
);

CREATE TABLE OFFICE
(
  ID_OFFICE NUMBER NOT NULL,
  NAME VARCHAR2(20) NOT NULL,
  EMAIL VARCHAR2(20) NOT NULL,
  PHONE VARCHAR2(20) NOT NULL,
  COUNTRY VARCHAR2(20) NOT NULL,
  CITY VARCHAR2(20) NOT NULL,
  STREET VARCHAR2(20) NOT NULL,
  CASE NUMBER
, CONSTRAINT OFFICE_PK PRIMARY KEY
  (
    ID_OFFICE
  )
  ENABLE
);

CREATE TABLE MODEL
(
  ID_MODEL NUMBER NOT NULL,
  MARK VARCHAR2(20) NOT NULL,
  MODEL VARCHAR2(20) NOT NULL,
  MAX_SPEED NUMBER NOT NULL,
  POWER NUMBER NOT NULL,
  TANK_VALUE NUMBER NOT NULL,
  GEAR_BOX VARCHAR2(20) NOT NULL,
  WEIGHT NUMBER NOT NULL,
  ID_MANUFACTURER NUMBER NOT NULL
, CONSTRAINT MODEL_PK PRIMARY KEY
  (
    ID_MODEL
  )
  ENABLE
);
```

```
CREATE TABLE MANUFACTURER
(
  ID_MANUFACTURER NUMBER NOT NULL,
  NAME VARCHAR2(20) NOT NULL
, CONSTRAINT MANUFACTURER_PK PRIMARY KEY
  (
    ID_MANUFACTURER
  )
  ENABLE
);

CREATE TABLE MANAGER
(
  ID_MANAGER NUMBER NOT NULL,
  FIRST_NAME VARCHAR2(20) NOT NULL,
  LAST_NAME VARCHAR2(20) NOT NULL,
  POST VARCHAR2(20) NOT NULL,
  INN VARCHAR2(20) NOT NULL,
  ID_OFFICE NUMBER NOT NULL
, CONSTRAINT MANAGER_PK PRIMARY KEY
  (
    ID_MANAGER
  )
  ENABLE
);


CREATE TABLE CAR
(
  ID_CAR NUMBER NOT NULL,
  PRICE NUMBER NOT NULL,
  YEAR NUMBER NOT NULL,
  COLOUR VARCHAR2(20) NOT NULL,
  ID_MODEL NUMBER NOT NULL,
  ID_OFFICE NUMBER NOT NULL
, CONSTRAINT CAR_PK PRIMARY KEY
  (
    ID_CAR
  )
  ENABLE
);


ALTER TABLE SALE
ADD CONSTRAINT IDCAR_UK UNIQUE
(
  ID_CAR
)
 ENABLE;
```

```
ALTER TABLE OFFICE
ADD CONSTRAINT NAME_UK UNIQUE
(
  NAME
)
 ENABLE;


ALTER TABLE MODEL
ADD CONSTRAINT MARK_MODEL_UK UNIQUE
(
  MODEL,
  MARK
)
 ENABLE;


ALTER TABLE MANAGER
ADD CONSTRAINT FIRSTNAME_LASTNAME_UK UNIQUE
(
  FIRST_NAME,
  LAST_NAME
)
 ENABLE;


ALTER TABLE SALE
ADD CONSTRAINT SALE_CAR_FK FOREIGN KEY
(
  ID_CAR
)
REFERENCES CAR
(
  ID_CAR
)
 ENABLE;


ALTER TABLE SALE
ADD CONSTRAINT SALE_MANAGER_FK FOREIGN KEY
(
  ID_MANAGER
)
REFERENCES MANAGER
(
  ID_MANAGER
)
 ENABLE;


ALTER TABLE MODEL
ADD CONSTRAINT MODEL_MANUFACTURER_FK FOREIGN KEY
(
  ID_MANUFACTURER
)
REFERENCES MANUFACTURER
(
  ID_MANUFACTURER
)
 ENABLE;


ALTER TABLE MANAGER
ADD CONSTRAINT MANAGER_OFFICE_FK FOREIGN KEY
(
  ID_OFFICE
)
REFERENCES OFFICE
(
  ID_OFFICE
)
 ENABLE;
```

```
ALTER TABLE CAR
ADD CONSTRAINT CAR_MODEL_FK FOREIGN KEY
(
  ID_MODEL
)
REFERENCES MODEL
(
  ID_MODEL
)
 ENABLE;

ALTER TABLE CAR
ADD CONSTRAINT CAR_OFFICE_FK FOREIGN KEY
(
  ID_OFFICE
)
REFERENCES OFFICE
(
  ID_OFFICE
)
 ENABLE;


CREATE SEQUENCE  "CAR_SEQ"  MINVALUE 0 MAXVALUE
1999 INCREMENT BY 1 START WITH 1000 NOCACHE;

CREATE SEQUENCE  "MANAGER_SEQ"  MINVALUE 0
MAXVALUE 2999 INCREMENT BY 1 START WITH 2000;

CREATE SEQUENCE  "MANUFACTURER_SEQ"  MINVALUE 0
MAXVALUE 3999 INCREMENT BY 1 START WITH 3000;

CREATE SEQUENCE  "MODEL_SEQ"  MINVALUE 0 MAXVALUE
4999 INCREMENT BY 1 START WITH 4000;

CREATE SEQUENCE  "OFFICE_SEQ"  MINVALUE 0 MAXVALUE
5999 INCREMENT BY 1 START WITH 5000;

CREATE SEQUENCE  "SALE_SEQ"  MINVALUE 0 MAXVALUE
6999 INCREMENT BY 1 START WITH 6000;
```