LÍNEAS

Acceso a Datos

Unai Esteibar Caballero
Iván Fernandes Lerín
David Gorriti Azkue
09/01/2015

LINEAS APP

```
package app;

import java.util.Scanner;

import app.accesslayer.LineasAccess;
import repo.validate.InputRepo;

public class LineasApp {

	public static void main(String[] args) {

		int opt = -1;

		while (opt!=0){
			opt = menu();
			chooseWay(opt);

		}

	}

	private static void chooseWay(int opt) {
		switch (opt){
		case 0: LineasAccess.endProgram();
		break;
		case 1: LineasAccess.addLinea();
		break;
		case 2: LineasAccess.updateLinea();
		break;
		case 3: LineasAccess.deleteLinea();
		break;
		case 4: LineasAccess.consultLinea();
		break;
		}
	}

	private static int menu() {
		Scanner sc = new Scanner(System.in);
		int opt;
		System.out.println(":::: LINEAS ::::\n");
		System.out.println("1. Añadir una linea");
		System.out.println("2. Modificar una linea");
		System.out.println("3. Borrar una linea");
```

```
		System.out.println("4. Buscar una linea");
		System.out.println("0. Salir");
		opt = InputRepo.askInt("Elija una opción", 0, 4);

		return opt;
	}

}
```

LINEASACCESS

```
package app.accesslayer;

import com.sun.corba.se.spi.orbutil.fsm.Input;

import repo.access.LineaDaoImpl;
import repo.objects.Linea;
import repo.validate.InputRepo;

public class LineasAccess {

	public static void endProgram() {
		System.out.println(":::: FIN DEL PROGRAMA ::::");

	}

	private static boolean validLineaNums(int lineNum, int albNum){
		LineaDaoImpl dbLineas = new LineaDaoImpl();

		Linea l = dbLineas.getLinea(lineNum, albNum);
		if (l==null){
			return true;
		}else{
			return false;
		}
	}

	public static void addLinea() {

		Linea line = new Linea();

		System.out.println(":: Introduzca la nueva línea ::");
		boolean lineaValid = false;
		int lineNum = 0;
```

```
		int albNum = 0;
		while(!lineaValid){
			lineNum = InputRepo.askInt("Número de línea");
			albNum = InputRepo.askInt("Número de albaran");
			if (validLineaNums(lineNum, albNum)){
				lineaValid = true;
			}else{
				System.out.println("Los datos introducidos no son válidos
porque esa linea ya existe. Inténtalo de nuevo.");
			}
		}

		line = askLineData();

		line.setAlbaran(albNum);
		line.setLinea(lineNum);

		LineaDaoImpl dbLineas = new LineaDaoImpl();

		dbLineas.insertLinea(line);

	}

	public static void updateLinea() {

		LineaDaoImpl dbLineas = new LineaDaoImpl();
		Linea line = new Linea();

		System.out.println(":: Introduzca la línea a borrar ::");
		boolean lineaValid = true;
		int lineNum = 0;
		int albNum = 0;
		while(lineaValid){
			lineNum = InputRepo.askInt("Número de línea");
			albNum = InputRepo.askInt("Número de albaran");
			if (!validLineaNums(lineNum, albNum)){
				lineaValid = false;
			}else{
				System.out.println("Los datos introducidos no son válidos
porque esa linea no existe. Inténtalo de nuevo.");
			}
		}

		showLinea(lineNum,albNum);
```

```
		line = askLineData();

		line.setLinea(lineNum);
		line.setAlbaran(albNum);

		dbLineas.updateLinea(line);

	}

	public static void deleteLinea(){

		System.out.println(":: Introduzca la línea a borrar ::");
		boolean lineaValid = true;
		int lineNum = 0;
		int albNum = 0;
		while(lineaValid){
			lineNum = InputRepo.askInt("Número de línea");
			albNum = InputRepo.askInt("Número de albaran");
			if (!validLineaNums(lineNum, albNum)){
				lineaValid = false;
			}else{
				System.out.println("Los datos introducidos no son válidos
porque esa linea no existe. Inténtalo de nuevo.");
			}
		}

		showLinea(lineNum,albNum);

		if (InputRepo.askBoolean("¿Borrar Linea?")){

			LineaDaoImpl dbLineas = new LineaDaoImpl();

			dbLineas.deleteLinea(lineNum,albNum);

		}else{

			System.out.println("Operación cancelada");

		}

	}

	public static void consultLinea(){

		System.out.println(":: Introduzca la línea a consultar ::");
```

```
		boolean lineaValid = true;
		int lineNum = 0;
		int albNum = 0;
		while(lineaValid){
			lineNum = InputRepo.askInt("Número de línea");
			albNum = InputRepo.askInt("Número de albaran");
			if (!validLineaNums(lineNum, albNum)){
				lineaValid = false;
			}else{
				System.out.println("Los datos introducidos no son válidos
porque esa linea no existe. Inténtalo de nuevo.");
			}
		}

		showLinea(lineNum,albNum);

	}

	public static void showLinea(int linea, int albaran){

		Linea line = new Linea();
		LineaDaoImpl dbLineas = new LineaDaoImpl();

		line = dbLineas.getLinea(linea, albaran);

		int artNum = line.getArticulo();
		int provNum = line.getProveedor();
		int cantidad = line.getCantidad();
		int descuento = line.getDescuento();
		double precio = line.getPrecio();

		System.out.println("Linea: " + linea);
		System.out.println("Albaran: " + albaran);
		System.out.println("Artículo: " + artNum);
		System.out.println("Proveedor: " + provNum);
		System.out.println("Cantidad: " + cantidad);
		System.out.println("Descuento: " + descuento);
		System.out.println("Precio: " + precio);

	}

	public static Linea askLineData(){

		Linea line = new Linea();
```

```
                int artNum = InputRepo.askInt("Número de artículo");
                int provNum = InputRepo.askInt("Número de proveedor");
                int cantidad = InputRepo.askInt("Cantidad");
                int descuento = InputRepo.askInt("Descuento");
                double precio = InputRepo.askDouble("Precio");

                line.setArticulo(artNum);
                line.setCantidad(cantidad);
                line.setDescuento(descuento);
                line.setPrecio(precio);
                line.setProveedor(provNum);

                return line;

        }

}
```

LINEADAOIMPL

```
package repo.access;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.RowMapper;
import org.springframework.jdbc.core.support.JdbcDaoSupport;

import repo.interfaces.LineaDao;
import repo.objects.Linea;

public class LineaDaoImpl extends JdbcDaoSupport implements LineaDao {

        public Linea getLinea(int lin, int alb) {

                StringBuffer sql = new StringBuffer();

                sql
                        .append(" SELECT *")
                        .append(" FROM ").append("lineas")
                        .append(" WHERE linea = ? AND albaran = ?");

                Object[] params = new Object[] { lin, alb };
```

```
		Linea li = (Linea) getJdbcTemplate().queryForObject( sql.toString(), params,
new ItemRowMapper());


		return li;
	}

	public void updateLinea(Linea linea) {
		StringBuffer sql = new StringBuffer();
		sql.append("UPDATE ").append("lineas").append(" SET precio = ?, ")
				.append("articulo = ?, ").append("albaran = ?, ")
				.append("cantidad = ?, ").append("proveedor = ?,
").append("descuento = ?")
				.append(" WHERE linea = ? and albaran = ?");

		Object[] params = new Object[] {

		linea.getPrecio(), linea.getArticulo(), linea.getAlbaran(), linea.getCantidad(),
linea.getProveedor(), linea.getDescuento(), linea.getLinea(), linea.getAlbaran() };

		getJdbcTemplate().update(sql.toString(), params);
	}

	public void insertLinea(Linea linea) {
		StringBuffer sql = new StringBuffer();

		sql.append("INSERT INTO ").append("lineas")
				.append(" (linea, precio, articulo, albaran, cantidad, proveedor,
descuento) ")
				.append("VALUES(?,?,?,?,?,?,?)");

		Object[] params = new Object[] { linea.getLinea(), linea.getPrecio(),
				linea.getArticulo(), linea.getAlbaran(), linea.getCantidad(),
linea.getProveedor(), linea.getDescuento()};

		getJdbcTemplate().update(sql.toString(), params);
	}

	public void deleteLinea(int linea, int alb) {

		StringBuffer sql = new StringBuffer();

		sql
		.append("DELETE FROM ").append("lineas")
		.append(" WHERE linea = ? AND albaran = ? ");
```

```
            Object[] params = new Object[] {linea, alb};

            getJdbcTemplate().update(sql.toString(), params);
        }

        class ItemRowMapper implements RowMapper {

            public Object mapRow(ResultSet rs, int index) throws SQLException {

                Linea li = new Linea();

                li.setLinea(rs.getInt("linea"));
                li.setArticulo(rs.getInt("articulo"));
                li.setAlbaran(new Integer(rs.getInt("albaran")));
                li.setCantidad(rs.getInt("cantidad"));
                li.setDescuento(rs.getInt("descuento"));
                li.setProveedor(rs.getInt("proveedor"));
                li.setPrecio(rs.getDouble("precio"));

                return li;
            }
        }
}
```

LINEADAO

```
package repo.interfaces;

import repo.objects.Linea;

public interface LineaDao {

        Linea getLinea(int linea, int alb);

        void updateLinea(Linea linea);

        void insertLinea(Linea linea);

        void deleteLinea(int linea, int alb);


}
```

LINEA

```
package repo.objects;

import java.io.Serializable;

public class Linea implements Serializable{

	//private static final long serialVersionUID = 1L;
	private int albaran, linea, articulo, proveedor, cantidad, descuento;
	private double precio;

	public int getAlbaran() {
		return albaran;
	}

	public void setAlbaran(int albaran) {
		this.albaran = albaran;
	}

	public int getLinea() {
		return linea;
	}

	public void setLinea(int linea) {
		this.linea = linea;
	}

	public int getArticulo() {
		return articulo;
	}

	public void setArticulo(int articulo) {
		this.articulo = articulo;
	}

	public int getProveedor() {
		return proveedor;
	}

	public void setProveedor(int proveedor) {
		this.proveedor = proveedor;
	}

	public int getCantidad() {
```

```
		return cantidad;
	}

	public void setCantidad(int cantidad) {
		this.cantidad = cantidad;
	}

	public int getDescuento() {
		return descuento;
	}

	public void setDescuento(int descuento) {
		this.descuento = descuento;
	}

	public double getPrecio() {
		return precio;
	}

	public void setPrecio(double precio) {
		this.precio = precio;
	}

	/*public static long getSerialversionuid() {
		return serialVersionUID;
	}*/

}
```

INPUTREPO

```
package repo.validate;

import java.util.InputMismatchException;
import java.util.Scanner;

public class InputRepo {

	public static int askInt(String msg, int min, int max) {
		Scanner sc = new Scanner(System.in);
		boolean correct = false;
		int res = 0;
		while (!correct) {
			System.out.print(msg + ": ");
```

```
                        try{
                                res = sc.nextInt();
                                if ((res >= min) && (res<=max)){
                                        return res;
                                }else{
                                        System.out.println("\nDebe introducir una opción
correcta. Vuelva a intentarlo.|n");
                                        sc.nextLine();
                                }

                        }catch(InputMismatchException e){
                                System.out.println("\nDebe introducir un número entero.
Vuelva a intentarlo.\n");
                                sc.nextLine();
                        }
                }
                return res;

        }


        public static String askString(String msg) {
                Scanner sc = new Scanner(System.in);
                System.out.print(msg + ": ");
                return sc.nextLine();
        }

        public static int askInt(String msg) {
                Scanner sc = new Scanner(System.in);
                boolean correct = false;
                int res = 0;
                while (!correct) {
                        System.out.print(msg + ": ");
                        try{
                                res = sc.nextInt();
                                return res;
                        }catch(InputMismatchException e){
                                System.out.println("\nDebe introducir un número entero.
Vuelva a intentarlo.\n");
                                sc.nextLine();
                        }
                }
                return res;

        }
```

```
	public static double askDouble(String msg) {
		Scanner sc = new Scanner(System.in);
		boolean correct = false;
		double res = 0;
		while (!correct) {
			System.out.print(msg + ": ");
			try{
				res = sc.nextDouble();
				return res;
			}catch(InputMismatchException e){
				System.out.println("\nDebe introducir un número entero.
Vuelva a intentarlo.\n");
				sc.nextLine();
			}
		}
		return res;

	}

	public static boolean askBoolean(String msg){

		String res = "";

		while (res.charAt(0) != 'S' && res.charAt(0) != 'N'){

			res = askString(msg).toUpperCase();

		}

		if (res.charAt(0) == 'S')
			return true;
		else
			return false;

	}

}
```

LINEADAOTEST

```
package tests;
```

```
import org.springframework.test.AbstractTransactionalDataSourceSpringContextTests;

import repo.interfaces.LineaDao;
import repo.objects.Linea;

@SuppressWarnings("deprecation")
public class LineaDaoTest extends
		AbstractTransactionalDataSourceSpringContextTests {

	private LineaDao lineaDao;

	public void setLineaDao(LineaDao lineaDao) {
		this.lineaDao = lineaDao;
	}

	protected String[] getConfigLocations() {
		this.setAutowireMode(AUTOWIRE_BY_NAME);
		return new String[] { "repo/context/Context-testLinea.xml" };
	}

	public void testInsertLinea() {
		// this.setDefaultRollback(false);
		Linea linea = new Linea();
		linea.setAlbaran(1);
		linea.setLinea(9999);
		linea.setArticulo(133);
		linea.setCantidad(5);
		linea.setDescuento(3);
		linea.setPrecio(300);
		linea.setProveedor(68);

		lineaDao.insertLinea(linea);

		Linea lineaResult = lineaDao.getLinea(linea.getLinea(), linea.getAlbaran());

		assertEquals(linea.getLinea(), lineaResult.getLinea());

	}

	public void testGetLinea() {
		Linea linea = new Linea();
		linea.setAlbaran(1);
		linea.setLinea(9999);
		linea.setArticulo(133);
		linea.setCantidad(5);
```

```
        linea.setDescuento(3);
        linea.setPrecio(300);
        linea.setProveedor(68);

        lineaDao.insertLinea(linea);

        Linea lineaResult = lineaDao.getLinea(linea.getLinea(), linea.getAlbaran());
        assertNotNull(lineaResult);
}


public void testUpdateLinea() {
        Linea linea = new Linea();
        linea.setAlbaran(1);
        linea.setLinea(9999);
        linea.setArticulo(133);
        linea.setCantidad(5);
        linea.setDescuento(3);
        linea.setPrecio(300);
        linea.setProveedor(68);

        lineaDao.insertLinea(linea);

        Linea linea2 = new Linea();
        linea2.setAlbaran(1);
        linea2.setLinea(9999);
        linea2.setArticulo(133);
        linea2.setCantidad(50);
        linea2.setDescuento(3);
        linea2.setPrecio(300);
        linea2.setProveedor(68);

        lineaDao.updateLinea(linea2);

        Linea lineaResult = lineaDao.getLinea(linea2.getLinea(), linea.getAlbaran());
        assertEquals(lineaResult.getCantidad(), 50);
}

public void testDeleteItem() {
        Linea linea = new Linea();
        linea.setAlbaran(1);
        linea.setLinea(9999);
        linea.setArticulo(133);
        linea.setCantidad(5);
        linea.setDescuento(3);
```

```
                linea.setPrecio(300);
                linea.setProveedor(68);

                lineaDao.insertLinea(linea);

                int lineaToDelete = 9999;
                int albaranToDelete = 1;
                lineaDao.deleteLinea(lineaToDelete, albaranToDelete);

                // si la ejecucion llega aqui significa que el delete se ha efectuado
                // correctamente
                assertTrue(true);

        }

}
```

CONTEXT

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:aop="http://www.springframework.org/schema/aop"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.0.xsd
            http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-2.0.xsd
            http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-2.0.xsd">

       <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"
              destroy-method="close">
              <property name="driverClassName" value="com.mysql.jdbc.Driver"
/>
              <property name="url" value="jdbc:mysql://localhost/almacen" />
              <property name="username" value="almacen" />
              <property name="password" value="almacen" />
       </bean>

       <bean id="transactionManager"

       class="org.springframework.jdbc.datasource.DataSourceTransactionManage
r">
              <property name="dataSource" ref="dataSource" />
       </bean>

       <bean id="abstractTiendaDao" abstract="true">
              <property name="dataSource" ref="dataSource" />
       </bean>
```

```
      <bean id="lineaDao" class="repo.access.LineaDaoImpl"
parent="abstractTiendaDao" />

</beans>
```