PROYECTO ACCESO A DATOS



Unai Arévalo y Mattin Irigoien

Índice

1. Planning.
   1. Conexión a la base de datos. Pág 3
   2. Creación del main y menú. Pág 3
   3. Lectura de XML. Pág 3
2. Diario. Pág 3-4
3. Código Fuente.
   1. Clase del objeto albaranes (Albaranes.java). Pág 5-6
   2. Interfaz del template (AlbaranesDao.java). Pág 6
   3. Clase que implementa la interfaz(AlbaranesJDBCTemplate). Pág 6-8
   4. Clase mapper (AlbaranesMapper.java). Pág 8
   5. Clase main (AlbaranesTest.java). Pág 8-14
   6. XML bean. Pág 14
   7. XML datos de la tabla. Pág 14

Planning:

-Conexión a la base de datos:

Lo primero que haremos será realizar la conexión a la base de datos por medio de SpringJDBC. Para ello crearemos un xml con beans para realizar la conexión. Después crearemos en java varias clases para realizar nuestra conexión con la tabla “Albaranes”. En total serán 4 clases imprescindibles: La de objeto, la interfaz, la que implementaremos la interfaz y la que implementaremos un mapper.

-Creacion del main y menú:

Finalmente, una quinta clase que será el main donde crearemos la conexión y gracias a las anteriores clases haremos las consultas precisan. En esta última clase habrá un menú para que el usuario decida que quiere hacer.

-Lectura del XML:

Para terminar, realizaremos varios métodos donde recogeremos datos de la tabla “Albaranes” y meteremos esos datos en un XML. Para ello usaremos el paquete de JavaDOM. Usando las mismas clases de JavaDOM leeremos el XML del grupo siguiente y meteremos esos datos en una nueva tabla.

Diario:

Día 0: Hemos creado el planning con todo lo que queremos hacer en este proyecto.

Día 1: Creamos las clases para la conexión de albaranes con el servidor (Albaranes, AlbaranesDao y AlbaranesJDBCTemplate).

Día 2: Creamos la clase AlbaranesMapper para recoger los datos después de hacer un “SELECT” e introducir los datos en un objeto Albaranes.

Día 3: Creamos el XML con el beans para la conexión con nuestra base de datos local.

Día 4: Empezamos a crear una clase main donde crearemos un menú con las opciones que se piden en el proyecto.

Día 5: Terminamos de crear el menú también con dos opciones de momento vacías, una para crear un XML con los datos de nuestra tabla (Albaranes) y otro para leer los datos del XML del grupo siguiente.

Día 6: Empezamos a hacer el programa que creará el XML, primero recogiendo los datos de la base de datos y después con clases de JavaDOM crearemos el XML. (Métodos: rellenarXML() y generateXML())

Día 7: Terminamos con los métodos para generar el XML y empezamos a crear otro para leer el XML del siguiente grupo. Hablamos con el grupo anterior para que su programa lea nuestro XML

Día 8: Terminamos el programa que leerá el XML del siguiente grupo. (Método: leerXML())

Día 9: Repasamos el programa en busca de errores y los solucionamos finalizando completamente el programa.

Día 10: Finalmente, hemos creado el informe dónde implantamos todas las clases de nuestro proyecto, el planning y este mismo diario.

CÓDIGO FUENTE:

3.1 Clase del objeto albaranes:

En esta clase creamos un objeto albaranes el cual tendrá como atributos todos los campos de la tabla albaranes a la que queremos conectarnos. También tendrá los métodos getters y setters por cada atributo para sus respectivas llamas y finalmente un método toString para devolver en un string toda la información de los atributos.

**package** almacen;

**public** **class** Albaranes {

**private** **int** albaran;

**private** **int** cliente;

**private** String fecha\_albaran;

**private** String fecha\_envio;

**private** String fecha\_pago;

**private** String formpago;

**private** String estado;

**public** **int** getAlbaran() {

**return** albaran;

}

**public** **int** getCliente() {

**return** cliente;

}

**public** String getFecha\_albaran() {

**return** fecha\_albaran;

}

**public** String getFecha\_envio() {

**return** fecha\_envio;

}

**public** String getFecha\_pago() {

**return** fecha\_pago;

}

**public** String getFormpago() {

**return** formpago;

}

**public** String getEstado() {

**return** estado;

}

**public** **void** setAlbaran(**int** albaran) {

**this**.albaran = albaran;

}

**public** **void** setCliente(**int** cliente) {

**this**.cliente = cliente;

}

**public** **void** setFecha\_albaran(String fecha\_albaran) {

**this**.fecha\_albaran = fecha\_albaran;

}

**public** **void** setFecha\_envio(String fecha\_envio) {

**this**.fecha\_envio = fecha\_envio;

}

**public** **void** setFecha\_pago(String fecha\_pago) {

**this**.fecha\_pago = fecha\_pago;

}

**public** **void** setFormpago(String formpago) {

**this**.formpago = formpago;

}

**public** **void** setEstado(String estado) {

**this**.estado = estado;

}

@Override

**public** String toString() {

**return** "Albaranes [albaran=" + albaran + ", cliente=" + cliente + ", fecha\_albaran=" + fecha\_albaran

+ ", fecha\_envio=" + fecha\_envio + ", fecha\_pago=" + fecha\_pago + ", formpago=" + formpago + ", estado="

+ estado + "]";

}

}

3.2 Interfaz del template:

En esta interfaz crearemos los métodos que vamos a usar para todas las consultas necesarios.

**package** almacen;

**import** java.util.List;

**import** javax.sql.DataSource;

**import** almacen.Albaranes;

**public** **interface** AlbaranesDAO {

**public** **void** setDataSource(DataSource ds);

**public** **void** actualizarAlbaran(**int** albaran, **int** cliente, String fecha\_albaran, String fecha\_envio, String fecha\_pago, String formpago, String estado);

**public** Albaranes selectAlbaran(**int** albaran);

**public** **void** borrarAlbaran(**int** albaran);

**public** **void** nuevoAlbaran(**int** cliente, String fecha\_albaran, String fecha\_envio, String fecha\_pago, String formpago, String estado);

**public** List<Albaranes> selectAllAlbaranes();

// Método cual haremos diferentes consultas

**public** **void** nuevaSQL(String sql);

}

3.3 Clase que implementa el mapper:

Esta clase implementará la interfaz que hemos creado antes con los métodos que vayamos a usar y aquí daremos forma a estos métodos. Uno de ellos, el “SELECT” requiere de un mapper para organizar toda la información a la que estas llamando para ello crearemos otro método que llamaremos AlbaranesMapper el cual creará un objeto Albaran.

**package** almacen;

**import** java.util.List;

**import** javax.sql.DataSource;

**import** org.springframework.jdbc.core.JdbcTemplate;

**import** almacen.Albaranes;

**public** **class** AlbaranesJDBCTemplate **implements** AlbaranesDAO{

**private** DataSource ds;

**private** JdbcTemplate template;

@Override

**public** **void** setDataSource(DataSource ds) {

**this**.ds = ds;

**this**.template = **new** JdbcTemplate(**this**.ds);

}

@Override

**public** **void** actualizarAlbaran(**int** albaran, **int** cliente, String fecha\_albaran, String fecha\_envio, String fecha\_pago, String formpago, String estado) {

String sql = "UPDATE albaranes SET cliente=?, fecha\_albaran=?, fecha\_envio=?, fecha\_pago=?, formpago=?, estado=? WHERE albaran=?";

template.update(sql, cliente, fecha\_albaran, fecha\_envio, fecha\_pago, formpago, estado, albaran);

System.***out***.println("Registro actualizado del ALBARAN = " + albaran);

}

@Override

**public** Albaranes selectAlbaran(**int** albaran) {

String sql = "SELECT \* FROM albaranes WHERE albaran = ?";

Albaranes al = template.queryForObject(sql, **new** Object[]{albaran}, **new** AlbaranesMapper());

**return** al;

}

@Override

**public** **void** borrarAlbaran(**int** albaran) {

String sql = "DELETE FROM albaranes WHERE albaran=?";

template.update(sql, albaran);

System.***out***.println("Registro eliminado: ALBARAN = " + albaran);

}

@Override

**public** **void** nuevoAlbaran(**int** cliente, String fecha\_albaran, String fecha\_envio, String fecha\_pago, String formpago, String estado) {

String sql = "INSERT INTO albaranes (albaran, cliente, fecha\_albaran, fecha\_envio, fecha\_pago, formpago, estado) VALUES ((SELECT MAX(albaran)+1 FROM(SELECT \* FROM albaranes) as X),?,?,?,?,?,?)";

template.update(sql, cliente, fecha\_albaran, fecha\_envio, fecha\_pago, formpago, estado);

System.***out***.println("Nuevo registro realizado");

}

@Override

**public** List<Albaranes> selectAllAlbaranes() {

List<Albaranes> al;

String sql = "SELECT \* FROM albaranes";

al = template.query(sql, **new** AlbaranesMapper());

**return** al;

}

@Override

**public** **void** nuevaSQL(String sql) {

template.update(sql);

}

}

3.4 Clase mapper:

Clase mapper necesaria para la consulta “SELECT”

**package** almacen;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** org.springframework.jdbc.core.RowMapper;

**import** almacen.Albaranes;

**public** **class** AlbaranesMapper **implements** RowMapper<Albaranes>{

@Override

**public** Albaranes mapRow(ResultSet rs, **int** arg1) **throws** SQLException {

Albaranes al = **new** Albaranes();

al.setAlbaran(rs.getInt("albaran"));

al.setCliente(rs.getInt("cliente"));

al.setFecha\_albaran(rs.getString("fecha\_albaran"));

al.setFecha\_envio(rs.getString("fecha\_envio"));

al.setFecha\_pago(rs.getString("fecha\_pago"));

al.setFormpago(rs.getString("formpago"));

al.setEstado(rs.getString("estado"));

**return** al;

}

}

3.5 Clase main:

En esta clase main lo primero que haremos será crear un menú por medio de un do-while y un switch para que el usuario decida que quiere hacer (seleccionar albarán, actualizar…).

Cada opción del menú hará una llamada a un método estático. Cada método hará su respectivo trabajo. Los principales de la tabla albarán (nuevoRegistro(), actualizarRegistro(), borrarRegistro() y consultarRegistro) les pasaremos como parámetro el objeto de AlbaranesJDBCTemplate que hemos creado el cual tendrá como métodos las consultas que queremos hacer y cada cual hará la que le corresponde.

Las otras dos opciones del menú serán para crear el XML y para leer el XML del siguiente grupo.

Para crear el XML realizaremos 2 métodos uno en el que realizaremos un “SELECT \*” de la tabla albaranes y guardaremos toda esa información en varios ArrayList, uno por cada campo. Estos ArrayList los pasaremos como parámetro al otro método y este se encargará de crear el XML y poner los datos que le hemos pasado.

Para terminar el último método será el encargado de recoger la información del XML del siguiente grupo. Para ello se importan clases y sigue un esquema parecido al método que crea el XML.

**package** almacen;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**import** org.w3c.dom.DOMImplementation;

**import** org.w3c.dom.Document;

**import** org.w3c.dom.Element;

**import** org.w3c.dom.Node;

**import** org.w3c.dom.NodeList;

**import** org.w3c.dom.Text;

**import** almacen.Albaranes;

**import** almacen.AlbaranesJDBCTemplate;

**import** java.io.File;

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.Scanner;

**import** javax.xml.parsers.DocumentBuilder;

**import** javax.xml.parsers.DocumentBuilderFactory;

**import** javax.xml.transform.Transformer;

**import** javax.xml.transform.TransformerFactory;

**import** javax.xml.transform.dom.DOMSource;

**import** javax.xml.transform.stream.StreamResult;

**public** **class** AlbaranesTest {

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.***in***);

**int** res;

// Conexión con el XML a la base de datos

ApplicationContext context = **new** ClassPathXmlApplicationContext("springjdbc.xml");

AlbaranesJDBCTemplate al = (AlbaranesJDBCTemplate) context.getBean("AlbaranesJDBCTemplate");

// Menu del programa

**do** {

System.***out***.println("MENU");

System.***out***.println("1. Nuevo registro");

System.***out***.println("2. Actualizar registro");

System.***out***.println("3. Borrar registro");

System.***out***.println("4. Consultar registro");

System.***out***.println("5. Rellenar XML");

System.***out***.println("6. Recoger XML");

System.***out***.println("7. Salir");

System.***out***.print("---->");

res = sc.nextInt();

**switch**(res) {

**case** 1:

*nuevoRegistro*(al);

**break**;

**case** 2:

*actualizarRegistro*(al);

**break**;

**case** 3:

*borrarRegistro*(al);

**break**;

**case** 4:

*consultarRegistro*(al);

**break**;

**case** 5:

*rellenarXML*(al);

**break**;

**case** 6:

*leerXML*(al);

**break**;

**case** 7:

System.***out***.println("Cerrando...");

**break**;

**default**:

**break**;

}

}**while**(res!=7);

System.***out***.println("Programa finalizado");

}

**public** **static** **void** nuevoRegistro(AlbaranesJDBCTemplate al) {

Scanner sc = **new** Scanner(System.***in***);

**int** cliente;

String fecha\_albaran, fecha\_envio, fecha\_pago, formpago, estado;

System.***out***.println("Introduce datos para nuevo registro:");

System.***out***.println("Código cliente:");

cliente = sc.nextInt();

sc.nextLine();

System.***out***.println("Fecha albaran (AAAA-MM-DD):");

fecha\_albaran = sc.nextLine();

System.***out***.println("Fecha albaran (AAAA-MM-DD):");

fecha\_envio = sc.nextLine();

System.***out***.println("Fecha albaran (AAAA-MM-DD):");

fecha\_pago = sc.nextLine();

System.***out***.println("Forma de pago (C/T)");

formpago = sc.nextLine();

System.***out***.println("Estado (S/N)");

estado = sc.nextLine();

al.nuevoAlbaran(cliente, fecha\_albaran, fecha\_envio, fecha\_pago, formpago, estado);

}

**public** **static** **void** actualizarRegistro(AlbaranesJDBCTemplate al) {

Scanner sc = **new** Scanner(System.***in***);

**int** albaran, cliente;

String fecha\_albaran, fecha\_envio, fecha\_pago, formpago, estado;

System.***out***.println("Introduce datos para actualizar registro:");

System.***out***.println("Código albaran para actualizar:");

albaran = sc.nextInt();

Albaranes nAlActu = al.selectAlbaran(albaran);

System.***out***.println(nAlActu.toString());

System.***out***.println("");

System.***out***.println("Introduce los nuevos datos:");

System.***out***.println("Código cliente:");

cliente = sc.nextInt();

sc.nextLine();

System.***out***.println("Fecha albaran (AAAA-MM-DD):");

fecha\_albaran = sc.nextLine();

System.***out***.println("Fecha albaran (AAAA-MM-DD):");

fecha\_envio = sc.nextLine();

System.***out***.println("Fecha albaran (AAAA-MM-DD):");

fecha\_pago = sc.nextLine();

System.***out***.println("Forma de pago (C/T)");

formpago = sc.nextLine();

System.***out***.println("Estado (S/N)");

estado = sc.nextLine();

al.actualizarAlbaran(albaran, cliente, fecha\_albaran, fecha\_envio, fecha\_pago, formpago, estado);

}

**public** **static** **void** consultarRegistro(AlbaranesJDBCTemplate al) {

Scanner sc = **new** Scanner(System.***in***);

**int** albaran;

System.***out***.println("Código de albaran para consulta:");

albaran = sc.nextInt();

Albaranes nAlb = al.selectAlbaran(albaran);

System.***out***.println(nAlb.toString());

}

**public** **static** **void** borrarRegistro(AlbaranesJDBCTemplate al) {

Scanner sc = **new** Scanner(System.***in***);

**int** albaran;

**char** confir;

System.***out***.println("Código de albaran para borrar:");

albaran = sc.nextInt();

Albaranes nAlBor = al.selectAlbaran(albaran);

System.***out***.println(nAlBor.toString());

**do** {

System.***out***.println("Seguro que quieres borrar este registro? (S/s / N/n)");

confir = sc.next().charAt(0);

} **while** (confir != 's' && confir != 'n' && confir != 'S' && confir != 'N');

**if** (confir == 's' || confir == 'S') {

al.borrarAlbaran(albaran);

} **else** {

System.***out***.println("No se ha borrado ningún registro.");

}

}

**public** **static** **void** rellenarXML(AlbaranesJDBCTemplate al) {

String nombre\_archivo = "albaranes";

ArrayList<String> albaranXML = **new** ArrayList<String>();

ArrayList<String> clienteXML = **new** ArrayList<String>();

ArrayList<String> fechaAlbaranXML = **new** ArrayList<String>();

ArrayList<String> fechaEnvioXML = **new** ArrayList<String>();

ArrayList<String> fechaPagoXML = **new** ArrayList<String>();

ArrayList<String> formpagoXML = **new** ArrayList<String>();

ArrayList<String> estadoXML = **new** ArrayList<String>();

List<Albaranes> listAlb = al.selectAllAlbaranes();

**for** (**int** i = 0; i < listAlb.size(); i++) {

albaranXML.add("" + listAlb.get(i).getAlbaran());

clienteXML.add("" + listAlb.get(i).getCliente());

fechaAlbaranXML.add(listAlb.get(i).getFecha\_albaran());

fechaEnvioXML.add(listAlb.get(i).getFecha\_envio());

fechaPagoXML.add(listAlb.get(i).getFecha\_pago());

formpagoXML.add(listAlb.get(i).getFormpago());

estadoXML.add(listAlb.get(i).getEstado());

}

**try** {

*generateXML*(nombre\_archivo, albaranXML, clienteXML, fechaAlbaranXML, fechaEnvioXML, fechaPagoXML, formpagoXML, estadoXML);

System.***out***.println("XML de albaranes generado");

} **catch** (Exception e) {}

}

**public** **static** **void** generateXML(String name, ArrayList<String> albaran, ArrayList<String> cliente, ArrayList<String> fechaAlbaran, ArrayList<String> fechaEnvio, ArrayList<String> fechaPago, ArrayList<String> formpago, ArrayList<String> estado) **throws** Exception {

DocumentBuilderFactory factory = DocumentBuilderFactory.*newInstance*();

DocumentBuilder builder = factory.newDocumentBuilder();

DOMImplementation implementation = builder.getDOMImplementation();

Document document = implementation.createDocument(**null**, name, **null**);

document.setXmlVersion("1.0");

Element root = document.getDocumentElement();

**for** (**int** i = 0; i < albaran.size(); i++) {

Element firstNode = document.createElement("albaran");

Element nodeAlb = document.createElement("albaran");

Text nodeAlbValue = document.createTextNode(albaran.get(i));

nodeAlb.appendChild(nodeAlbValue);

Element nodeCli = document.createElement("cliente");

Text nodeCliValue = document.createTextNode(cliente.get(i));

nodeCli.appendChild(nodeCliValue);

Element nodeFecAl = document.createElement("fecha\_albaran");

Text nodeFecAlValue = document.createTextNode(fechaAlbaran.get(i));

nodeFecAl.appendChild(nodeFecAlValue);

Element nodeFecEn = document.createElement("fecha\_envio");

Text nodeFecEnValue = document.createTextNode(fechaEnvio.get(i));

nodeFecEn.appendChild(nodeFecEnValue);

Element nodeFecPag = document.createElement("fecha\_pago");

Text nodeFecPagValue = document.createTextNode(fechaPago.get(i));

nodeFecPag.appendChild(nodeFecPagValue);

Element nodeFormp = document.createElement("formpago");

Text nodeFormpValue = document.createTextNode(formpago.get(i));

nodeFormp.appendChild(nodeFormpValue);

Element nodeEst = document.createElement("estado");

Text nodeEstValue = document.createTextNode(estado.get(i));

nodeEst.appendChild(nodeEstValue);

firstNode.appendChild(nodeAlb);

firstNode.appendChild(nodeCli);

firstNode.appendChild(nodeFecAl);

firstNode.appendChild(nodeFecEn);

firstNode.appendChild(nodeFecPag);

firstNode.appendChild(nodeFormp);

firstNode.appendChild(nodeEst);

root.appendChild(firstNode);

}

TransformerFactory transformerFactory = TransformerFactory.*newInstance*();

Transformer transformer = transformerFactory.newTransformer();

DOMSource source = **new** DOMSource(document);

StreamResult result = **new** StreamResult(**new** File("C:/Users/adminportatil/workspace/AaD - Proyecto/src/listalbaranes.xml"));

transformer.transform(source, result);

}

**public** **static** **void** leerXML(AlbaranesJDBCTemplate al) {

String table\_name = "PROYECTO\_AaD\_2DM3\_GRUPO2";

**try** {

String sql="CREATE TABLE " + table\_name + " (cliente int(3), empresa varchar(25), apellidos varchar(25), nombre varchar(25), direccion1 varchar(25), direccion2 varchar(25), poblacion varchar(25), provincia int(2), distrito int(5), telefono varchar(25), formpago varchar(25), total\_factura decimal(11,2))";

al.nuevaSQL(sql);

System.***out***.println("Nueva tabla creada!");

File fXmlFile = **new** File("C:/Users/adminportatil/workspace/AaD - Proyecto/src/listgrupo3.xml"); // Aqui iría la ruta del xml del siguiente grupo

DocumentBuilderFactory lecFactory = DocumentBuilderFactory.*newInstance*();

DocumentBuilder lecBuilder = lecFactory.newDocumentBuilder();

Document lecDoc = lecBuilder.parse(fXmlFile);

lecDoc.getDocumentElement().normalize();

NodeList nList = lecDoc.getElementsByTagName("cliente");

**for** (**int** i = 0; i < nList.getLength(); i++) {

Node nNode = nList.item(i);

**if** (nNode.getNodeType() == Node.***ELEMENT\_NODE***) {

Element el = (Element) nNode;

String elCli = el.getElementsByTagName("cliente").item(0).getTextContent() + ", '";

String elEm = el.getElementsByTagName("empresa").item(0).getTextContent() + "', '";

String elAp = el.getElementsByTagName("apellidos").item(0).getTextContent() + "', '";

String elNom = el.getElementsByTagName("nombre").item(0).getTextContent() + "', '";

String elDir1 = el.getElementsByTagName("direccion1").item(0).getTextContent() + "', '";

String elDir2 = el.getElementsByTagName("direccion2").item(0).getTextContent() + "', '";

String elPob = el.getElementsByTagName("poblacion").item(0).getTextContent() + "', ";

String elPro = el.getElementsByTagName("provincia").item(0).getTextContent() + ", ";

String elDis = el.getElementsByTagName("distrito").item(0).getTextContent() + ", '";

String elTel = el.getElementsByTagName("telefono").item(0).getTextContent() + "', '";

String elFor = el.getElementsByTagName("formpago").item(0).getTextContent() + "', ";

String elTot = el.getElementsByTagName("total\_factura").item(0).getTextContent();

String sql2 = "INSERT INTO " + table\_name + " VALUES (" + elCli + elEm + elAp + elNom + elDir1 + elDir2 + elPob + elPro + elDis + elTel + elFor + elTot + ")";

al.nuevaSQL(sql2);

}

}

System.***out***.println("Datos añadidos a la nueva tabla!");

} **catch**(Exception e) {

System.***out***.println("ERROR:" + e);

}

}

}

3.6 XML bean:

Este es el XML que necesitamos para realizer la conexión a la base de datos.

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

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd"*>

<bean id=*"dataSource"*

class=*"org.springframework.jdbc.datasource.DriverManagerDataSource"*>

<property name=*"driverClassName"* value=*"com.mysql.jdbc.Driver"*/>

<property name=*"url"* value=*"jdbc:mysql://localhost/almacen"*/>

<property name=*"username"* value=*"root"*/>

<property name=*"password"* value=*"050196"*/>

</bean>

<bean id=*"AlbaranesJDBCTemplate"*

class=*"almacen.AlbaranesJDBCTemplate"*>

<property name=*"dataSource"* ref=*"dataSource"* />

</bean>

</beans>

3.7 XML datos de la tabla:

Este es el XML que creamos con el mismo programa gracias a los métodos explicados más arriba.

\*\*Adjunto el programa y ahí se encuentra el XML, al copiar sale mal y no se entiende así que creo que mejor no copiarlo