Licencias Locales JAVA

20/11/2018

Tihomir Stoychev Stoychev
entrenamientovikingo.com
C MESTRE SERRANO 4 3 9
46650 CANALS (VALENCIA)

Visión general

Este proyecto hecho en JAVA será una gestion del ayuntamiento de Zaragoza, mas concretamente de las licencias que sus locales tienen , procesando todos sus datos necesarios para generar objetos a partir de clases y el proceso viceversa, a partir de objetos a XML.

Objetivos

1. Conseguir leer a partir de un xml y generar el arbol de nodos.
2. Al tener el arbol de nodos, instanciar las clases correspondientes con datos obtenidos del XML leido.
3. Crear un nuevo arbol de nodos desde 0 y vacio, para posteriori, ser rellenado con datos que le vienen de instancias de clase ejemplares.
4. Guardarlo en un documento XML con las propiedades de Indent.
5. Guardar un XML a BBDD

Tecnologias y librerias usadas

A continuación se van a describir las tecnologias usadas para llevar a cabo este proyecto y a medida que se usen nuevas, serán añadidas en este apartado.

Netbeans

IDE, que facilita la programación en JAVA. En su versión estable 8.2.

Git CLI

Git Comand Line, son los comandos necesarios para llevar el control de versiones del proyecto, aplicados desde terminal y usando instrucciones basicas como : clone,push,pull y merge.

Mysql Connector

Es la libreria requeria para poder conectarse a nuestra base de datos creada en phpmyadmin.

1.- XML

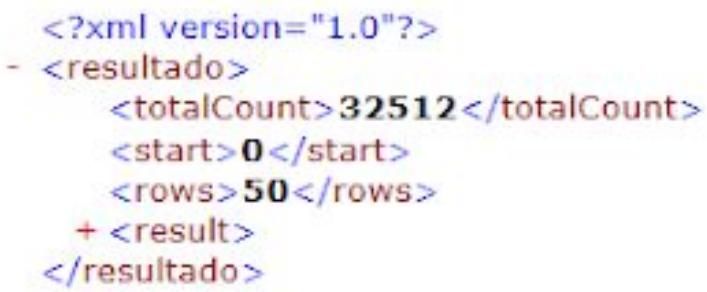
La fuente

Ayuntamiento de Zaragoza, gestion de Licencias de locales.

https://www.zaragoza.es/sede/servicio/registro-licencia.xml

La raiz

El elemento padre se llama **resultado**. Este tiene algunos hijos que son meramente estadisticos. El apartado rows por ejemplo nos dice las filas que tiene este XML. Por lo que con este XML estariamos procesando 50 locales con licecias (muchas) cada uno. Dentro de la etiqueta **resultado** esta la etiqueta **result** que contiene todos los locales como hijos.



Etiqueta <local>

Esta etiqueta es un hijo directo de **result**. La etiqueta **local** a su vez contiene hijos como: id,emplazamiento,codPortal,numVia... Que nos interesan para leerlas más tarde y rellenar las instancias de clase en JAVA. La etiqueta **local** tiene a su vez una etiqueta hija llamada **iae,** que no será necesaria procesar y será ignorada.

```
- <licencias>
  - <licencia>
    - <id>
      - <iae>
          <expediente>1086634</expediente>
          <anyo>2010</anyo>
        </iae>
      </id>
    - <tipo>
      - <tipo>
          <id>16</id>
          <title>Cambio de Titularidad de Licencia Funcionamiento</title>
          <creationDate>2009-03-17T00:00:00Z</creationDate>
        </tipo>
      </tipo>
      <orden>5</orden>
      <idResolucion>0</idResolucion>
      <resolucion>2011-11-03T00:00:00Z</resolucion>
      <creationDate>2011-11-29T00:00:00Z</creationDate>
      <lastUpdated>2011-11-29T00:00:00Z</lastUpdated>
    </licencia>
  - <licencia>
    - <id>
      - <iae>
          <expediente>1103257</expediente>
          <anyo>2007</anyo>
        </iae>
      </id>
    - <tipo>
      - <tipo>
          <id>16</id>
          <title>Cambio de Titularidad de Licencia Funcionamiento</title>
          <creationDate>2009-03-17T00:00:00Z</creationDate>
        </tipo>
      </tipo>
      <orden>3</orden>
      <idResolucion>0</idResolucion>
      <comments>Con suspensión cautelar</comments>
      <resolucion>2008-11-17T00:00:00Z</resolucion>
      <creationDate>2009-11-20T00:00:00Z</creationDate>
      <lastUpdated>2009-11-20T00:00:00Z</lastUpdated>
    </licencia>
  - <licencia>
    - <id>
      - <iae>
          <expediente>1351931</expediente>
          <anyo>2010</anyo>
        </iae>
      </id>
    - <tipo>
      - <tipo>
          <id>0</id>
          <title>Otras</title>
          <creationDate>2007-04-01T00:00:00Z</creationDate>
        </tipo>
      </tipo>
      <orden>4</orden>
      <idResolucion>0</idResolucion>
      <comments>Queda enterado de la adaptación al Catálogo de Espectáculos Públicos, A
      <resolucion>2010-11-09T00:00:00Z</resolucion>
```


Etiqueta <licencias>

Esta etiqueta es hija de **local,** siguiendo la logica de " un local tiene muchas licencias". De la licencia nos interesa conocer los datos de las etiquetas padre **id** y **tipo,** adentrandonos en sus hijos para obtener dichos dato. Aparte de la etiqueta licencias nos interesa capturar el dato de la etiqueta **<creationDate>.**

2.- JAVA

Las clases son usadas como plantillas para organizar los datos que vienen de los archivos XML y estructurarlos.

Clase Resultado

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package modelo;

import java.util.ArrayList;
import java.util.Scanner;

/**
 *
 * @author sportak
 */
public class Resultado extends ArrayList<Local> {

    public Resultado() {
    }

    public int getModCount() {
        return modCount;
    }

    public void setModCount(int modCount) {
        this.modCount = modCount;
    }

    public void impresionTOTAL() {
        Scanner entrada = new Scanner("empresa.xml");
        System.out.println("El archivo es ");
        while (entrada.hasNextLine()) {
            System.out.println(entrada.nextLine());
        }
        System.out.println("Su arbol de nodos es");

        for (int i = 0; i < this.size(); i++) {
            System.out.println(this.get(i).toString());
        }
    }

    @Override
    public String toString() {
        return "Resultado{" + '}';
    }

}

}
```

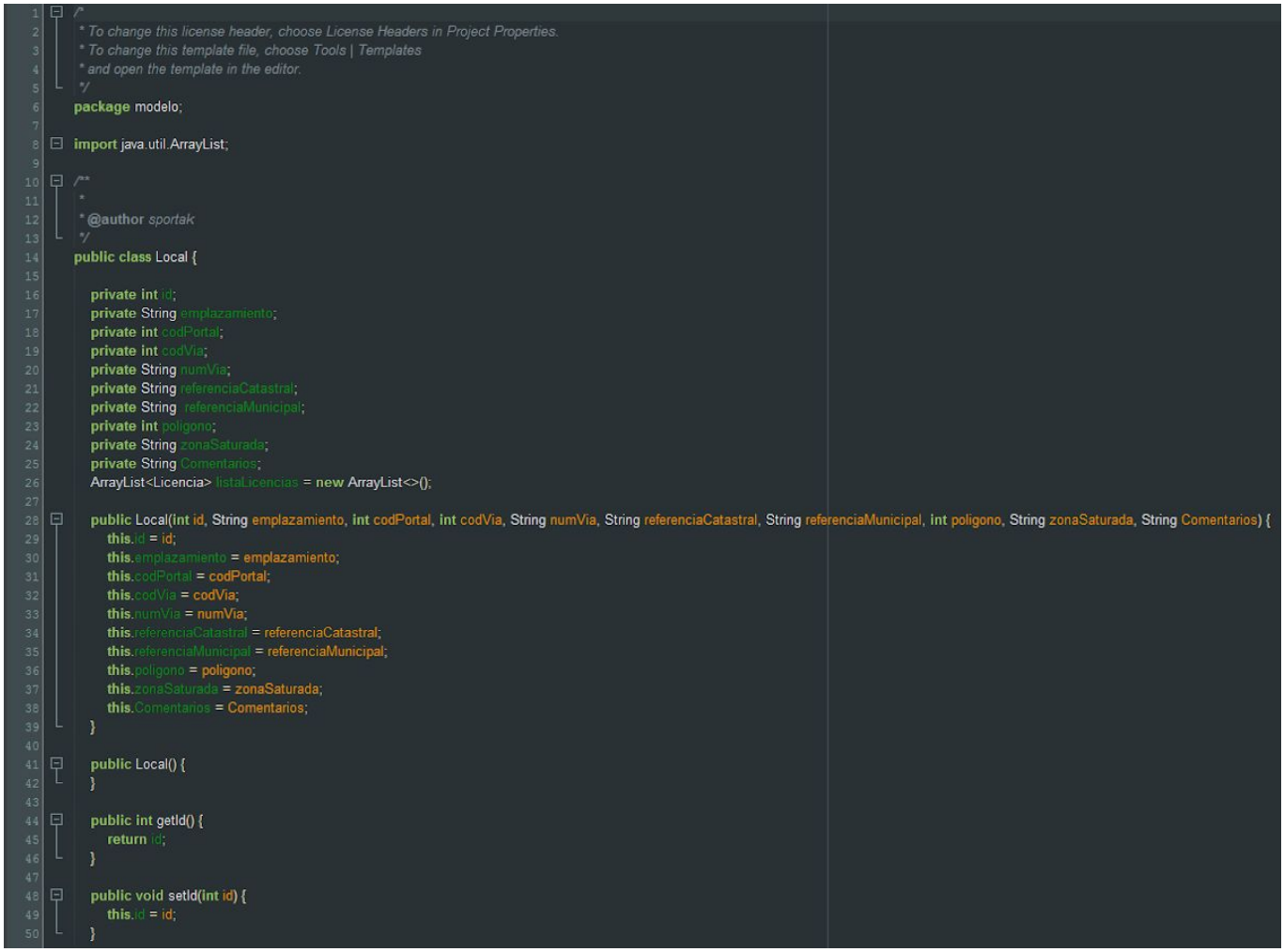
Esta clase hereda/extiende de ArrayList<Local>, por lo que funciona como un arraylist normal con sus metodos rutinarios como el .add() .removeAt() ... Aparte del constructor y algunos getters predefinidos tiene la impresión total, que dado un objeto **Resultado** se imprime su contenido como fichero y los datos de sus **hijos**.

Clase Licencia

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package modelo;

/**
 *
 * @author sportak
 */
public class Licencia {
    private String expediente;
    private String anyo;
    private int id;
    private String titulo;
    private String fechaCreacion;

    public Licencia(String expediente, String anyo, int id, String titulo, String fechaCreacion) {
        this.expediente = expediente;
        this.anyo = anyo;
        this.id = id;
        this.titulo = titulo;
        this.fechaCreacion = fechaCreacion;
    }
    public Licencia(){}

    public String getExpediente() {
        return expediente;
    }

    public void setExpediente(String expediente) {
        this.expediente = expediente;
    }

    public String getAnyo() {
        return anyo;
    }

    public void setAnyo(String anyo) {
        this.anyo = anyo;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

}
```

La clase **Licencia** alberga varios atributos de tipo privado, junto a dos constructores para cada situación. Aparte de ello cuenta con los tipicos getters y setters aparte del toString()

Clase Local



```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package modelo;

import java.util.ArrayList;

/**
 *
 * @author sportak
 */
public class Local {

    private int id;
    private String emplazamiento;
    private int codPortal;
    private int codVia;
    private String numVia;
    private String referenciaCatastral;
    private String  referenciaMunicipal;
    private int poligono;
    private String zonaSaturada;
    private String Comentarios;
    ArrayList<Licencia> listaLicencias = new ArrayList<>();

    public Local(int id, String emplazamiento, int codPortal, int codVia, String numVia, String referenciaCatastral, String referenciaMunicipal, int poligono, String zonaSaturada, String Comentarios) {
        this.id = id;
        this.emplazamiento = emplazamiento;
        this.codPortal = codPortal;
        this.codVia = codVia;
        this.numVia = numVia;
        this.referenciaCatastral = referenciaCatastral;
        this.referenciaMunicipal = referenciaMunicipal;
        this.poligono = poligono;
        this.zonaSaturada = zonaSaturada;
        this.Comentarios = Comentarios;
    }

    public Local() {
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }
```

La clase **Local** se lleva la peor parte y tiene muchos atributos, todos privados. Como distinción, tiene un arrayList de Licencias. Aparte de los constructores tiene los tipicos getters y setters acompañados del toString().

Clase Constantes (controlador.Constantes)

```java
package controlador;

/**
 *
 * @author sportak
 */
public class Constantes {

    public static String ET_RESULT = "result";
    public static String ET_ID = "id";
    public static String ET_EMPLAZAMIENTO = "emplazamiento";
    public static String ET_CODPORTAL = "codPortal";
    public static String ET_CODVIA = "codVia";
    public static String ET_NUMVIA = "numVia";
    public static String ET_REFCAT = "referenciaCatastral";
    public static String ET_REFMUN = "referenciaMunicipal";
    public static String ET_POLI = "poligono";
    public static String ET_ZSAT = "zonaSaturada";
    public static String ET_COMMENTS = "comments";
    public static String ET_LICENCIAS = "licencias";
    public static String ET_LICENCIA="licencia";
    public static String ET_LICENCIASIAE = "iae";
    public static String ET_LICENCIASTIPO = "tipo";
    public static String ET_EXPEDIENTE = "expediente";
    public static String ET_ANYO = "anyo";
    public static String ET_TIPO = "tipo";
    public static String ET_TITULO="title";
    public static String ET_CREATIONDATE="creationDate";
}
```

La clase **Constantes** tiene un uso muy especifico, y es facilitar la vida a los programadores que se olvidan rapido o no quieren perder tiempo buscando los nombres de las etiquetas en todo momento. Es la “biblia” donde se guardan los nombres de las etiquetas de forma **estatica,** por lo que se puede llamar desde cualquier clase con **Constantes.NombreVar.**

Clase ControlDom (controlador.ControlDom)

```java
public class ControlDom {

    public Document deXMLaDOC(File archivo) throws ParserConfigurationException, SAXException, IOException {
        Document doc = null;
        doc = DocumentBuilderFactory.newInstance().newDocumentBuilder().parse(archivo);
        return doc;
    }

    public Document deXMLaDOC() throws ParserConfigurationException {
        Document doc = null;
        doc = DocumentBuilderFactory.newInstance().newDocumentBuilder().newDocument();
        return doc;
    }

    //De DOM a XML
    public void deDOCaXML(Document doc, File file) throws TransformerConfigurationException, TransformerException {
        Transformer trans = TransformerFactory.newInstance().newTransformer();
        trans.setOutputProperty(OutputKeys.INDENT, "yes"); //indentar XML

        StreamResult result = new StreamResult(file);
        DOMSource source = new DOMSource(doc);
        trans.transform(source, result);
    }

    //Obtener ETIQUETAS
    public static Element getElementEtiqueta(String ETIQUETA, Element elemento) {
        return (Element) elemento.getElementsByTagName(ETIQUETA).item(0);
    }

    //Obtener VALOR de las etiquetas
    public static String getValorEtiqueta(String ETIQUETA, Element elemento) {
        Node nValue = elemento.getElementsByTagName(ETIQUETA).item(0);
        return nValue.getChildNodes().item(0).getNodeValue();
    }

    //Obtener ATRIBUTO
    public static String getAtributoEtiqueta(Element elemento, String ETIQUETA) {
        return elemento.getAttribute(ETIQUETA);
    }

}
```

La clase **ControlDom** es la encargada de hacer los trabajos basicos y reutilizables en todas sus clases hereditarias.

deXMLaDOC(): Se encarga de generar un arbol de nodos a partir de un fichero proporcionado o si no se proporciona uno, generar un arbol de nodos en blanco.

deDOCaXML(): Una vez se tenga un arbol de nodos DOM se puede guardar a un fichero XML, es lo que hace este metodo. Tambien cabe destacar el metodo setOutputProperty() que realiar la correcta organización del codigo XML.

getElementEtiqueta(), getValorEtiqueta(), getAtributoEtiqueta(): Son los metodos que usaremos para obtener un valor a partir de una etiqueta padre o un elemento de una etiqueta padre. De esta forma nos ahorramos mucho tiempo.

Clase ControlLocal (controlador.ControlLocal)

```
    public void escribirLocal(Document doc, Element result, Local local) {
        Element eleLocal = doc.createElement("local");
        result.appendChild(eleLocal);

        Element id = doc.createElement(Constantes.ET_ID);
        id.appendChild(doc.createTextNode(local.getId() + ""));
        eleLocal.appendChild(id);

        Element emplazamiento = doc.createElement(Constantes.ET_EMPLAZAMIENTO);
        emplazamiento.appendChild(doc.createTextNode(local.getEmplazamiento()));
        eleLocal.appendChild(emplazamiento);

        Element codPortal = doc.createElement(Constantes.ET_CODPORTAL);
        codPortal.appendChild(doc.createTextNode(local.getCodPortal() + ""));
        eleLocal.appendChild(codPortal);

        Element codVia = doc.createElement(Constantes.ET_CODVIA);
        codVia.appendChild(doc.createTextNode(local.getCodVia() + ""));
        eleLocal.appendChild(codVia);

        Element numVia = doc.createElement(Constantes.ET_NUMVIA);
        numVia.appendChild(doc.createTextNode(local.getNumVia()));
        eleLocal.appendChild(numVia);

        Element referenciaCatastral = doc.createElement(Constantes.ET_REFCAT);
        referenciaCatastral.appendChild(doc.createTextNode(local.getReferenciaCatastral()));
        eleLocal.appendChild(referenciaCatastral);

        Element referenciaMunicipal = doc.createElement(Constantes.ET_REFMUN);
        referenciaMunicipal.appendChild(doc.createTextNode(local.getReferenciaMunicipal()));
        eleLocal.appendChild(referenciaMunicipal);

        Element poligono = doc.createElement(Constantes.ET_POLI);
        poligono.appendChild(doc.createTextNode(local.getPoligono() + ""));
        eleLocal.appendChild(poligono);

        Element zonaSaturada = doc.createElement(Constantes.ET_ZSAT);
        zonaSaturada.appendChild(doc.createTextNode(local.getZonaSaturada() + ""));
        eleLocal.appendChild(zonaSaturada);

        Element comentarios = doc.createElement(Constantes.ET_COMMENTS);
        comentarios.appendChild(doc.createTextNode(local.getComentarios() + ""));
        eleLocal.appendChild(comentarios);

        escribirLicencias(eleLocal, doc, local);

    }
```

```
    private void escribirLicencias(Element eleLocal, Document doc, Local local) {
        Element licencias = doc.createElement(Constantes.ET_LICENCIAS);
        eleLocal.appendChild(licencias);

        for (int i = 0; i < local.getListaLicencias().size(); i++) {
            Element licencia = doc.createElement(Constantes.ET_LICENCIA);
            licencias.appendChild(licencia);

            Element id = doc.createElement(Constantes.ET_ID);
            licencia.appendChild(id);

            Element iae = doc.createElement(Constantes.ET_LICENCIASIAE);
            id.appendChild(iae);

            Element expediente = doc.createElement(Constantes.ET_EXPEDIENTE);
            expediente.appendChild(doc.createTextNode(local.getListaLicencias().get(i).getExpediente()));
            iae.appendChild(expediente);

            Element anyo = doc.createElement(Constantes.ET_ANYO);
            anyo.appendChild(doc.createTextNode(local.getListaLicencias().get(i).getAnyo()));
            iae.appendChild(anyo);

            Element tipo = doc.createElement(Constantes.ET_TIPO);
            licencia.appendChild(tipo);

            Element tipo2 = doc.createElement(Constantes.ET_TIPO);
            tipo.appendChild(tipo2);

            Element id2 = doc.createElement(Constantes.ET_ID);
            id2.appendChild(doc.createTextNode(local.getListaLicencias().get(i).getId() + ""));
            tipo2.appendChild(id2);

            Element titulo = doc.createElement(Constantes.ET_TITULO);
            titulo.appendChild(doc.createTextNode(local.getListaLicencias().get(i).getTitulo()));
            tipo2.appendChild(titulo);

            Element fechaCreacion = doc.createElement(Constantes.ET_CREATIONDATE);
            fechaCreacion.appendChild(doc.createTextNode(local.getListaLicencias().get(i).getFechaCreacion()));
            tipo2.appendChild(fechaCreacion);

        }

    }
}
```

La clase **ControlLocal** hace 2 simples funciones, **leer y escribir.** A continuación se propone describir en mayor detalle el funcionamiento.

leerLocal(Element elementoLocal)

Se crea un nuevo objeto instancia de la clase Local , sin parametros, el cual apartir de un element, se le iran rellenando los datos con unos simples setters. En un momento dato se comprueba si existen hijos con el nombre “comments” ya que es opcional y se debe controlar para evitar errores de ejecución. Tambien a la hora de leer el arrayList se llama a otro metodo el cual se explicará a continuación.

leerLicencia(Element elementoLocal,Local local)

A esta funcion se le pasa de nuevo el elemento Local y el objeto local en si que será donde se guardará el arraylist. Despues, se llama al metodo getElementsByTagName() pasandole

como referencia la etiqueta de las licencias , el cual devuelve todas las etiquetas con el nombre “licencias”. Esas etiquetas se guardan en un nodelist para más tarde ser recorrido y por cada item leido y generado un objeto Licencia que posteriormente se guarda en el arrayList “listaLicencias” en cada Local.

escribirLicencia(Document doc , Element result, Local loc)

Por cada elemento resultado se crea un local con sus licencias, por lo que basicamente se hace en esta funcion es crear un elemento Local y apartir del del objeto **loc** ir cogiendo sus datos y con ellos rellenar el elemento creado. A la hora de de escribir las licencias del local se llama al siguiente metodo.

escribirLicencias(Document doc , Element result, Local loc)

Por cada licencia que tiene el objeto Local se recorren y se van añadiendo como hijos del elemento “licencias”.

Clase ControlResultado (controlador.ControlResultado)

```java
/**
 *
 * @author sportak
 */
public class ControlResultado extends ControlDom {

    public Document recuperar(File xmlFile) throws ParserConfigurationException, SAXException, IOException {
        Document doc = null;
        doc = deXMLaDOC(xmlFile);
        return doc;
    }

    public void almacenar(Document doc, File archivoDestino) throws TransformerException {
        deDOCaXML(doc, archivoDestino);
    }


    public Resultado leerResult(Document doc) {
        ControlLocal cl = new ControlLocal();
        Element raizResultado = doc.getDocumentElement();
        Element etiquetaResult = getElementEtiqueta(Constantes.ET_RESULT, raizResultado);
        NodeList listaLocales = etiquetaResult.getChildNodes();
        Resultado res = new Resultado();

        for (int i = 0; i < listaLocales.getLength(); i++) {
            if (listaLocales.item(i).getNodeType() == Node.ELEMENT_NODE) {
                System.out.println("Voy a tramitar " + listaLocales.item(i).getNodeName() + " " + listaLocales.item(i).getChildNodes().getLength());
                res.add(cl.leerLocal((Element) listaLocales.item(i)));
            }
        }
        return res;
    }

    public void escribirResult(Document doc, Resultado res) {
        ControlLocal cl = new ControlLocal();

        Element raiz = doc.createElement("resultado");
        doc.appendChild(raiz);

        Element result = doc.createElement("result");
        raiz.appendChild(result);

        for (int i = 0; i < res.size(); i++) {
            cl.escribirLocal(doc, result, res.get(i));
        }

    }

}
```

recuperar(File f)

Dado un **fichero o File ,** este metodo **devuelve un document** cargado sobre ese fichero.

almacenar(Document doc, File f)

Dados un fichero y un doc, este ultimo guarda en el fichero destino.

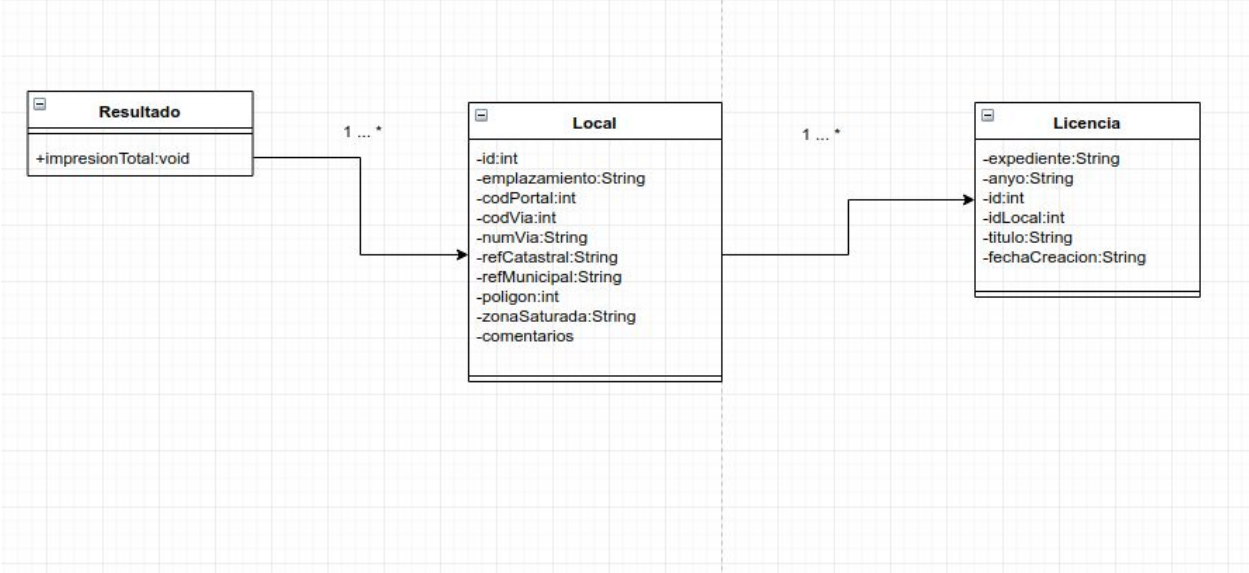
leerResult(Document doc)

A partir de un documento de nodos se comienza la lectura inicial

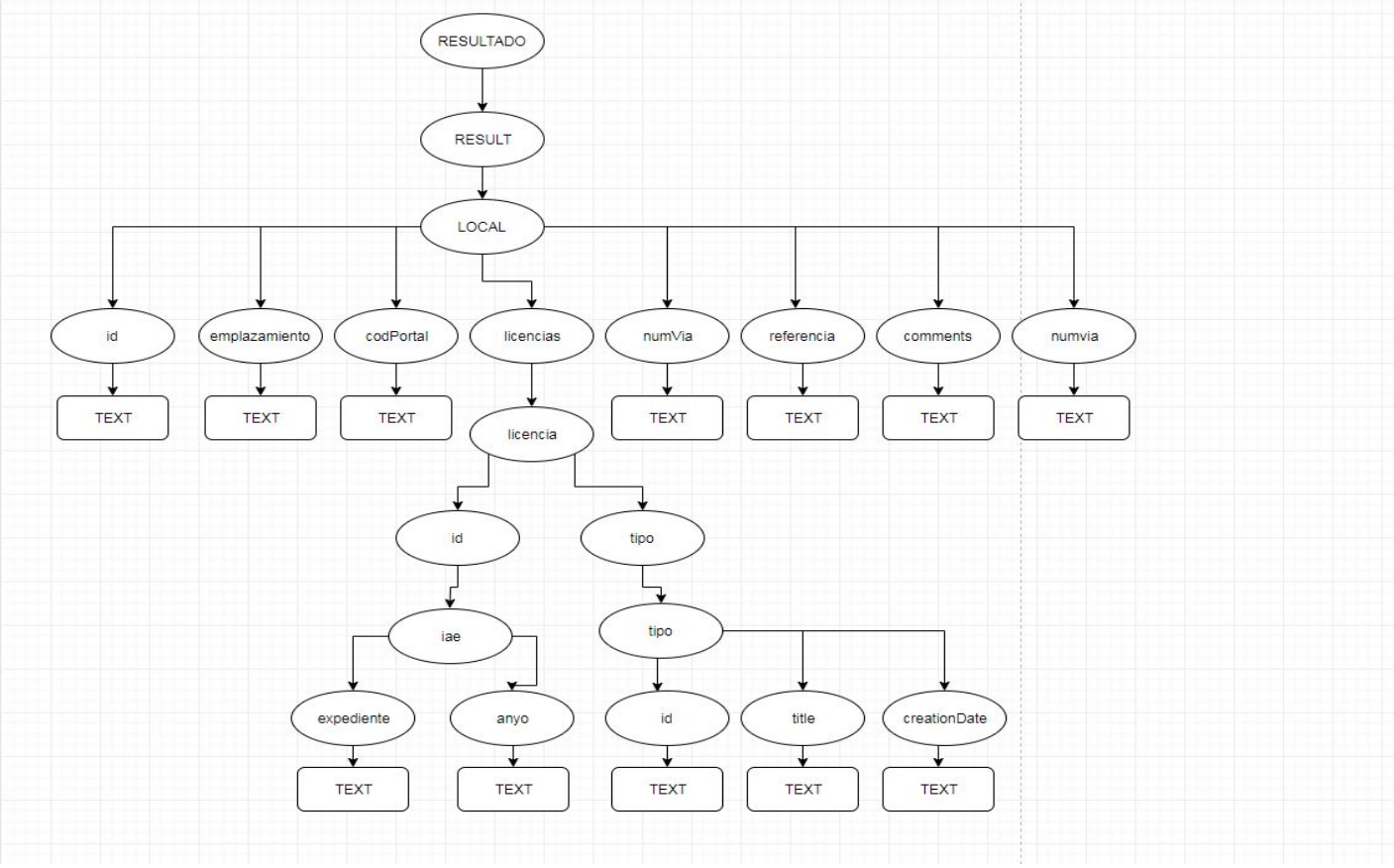
escribirResult(Document doc)

A partir de un documento vacio. Se le añaden como hijos los datos del objeto ControlLocal.

UML de las clases basicas (POJOS)



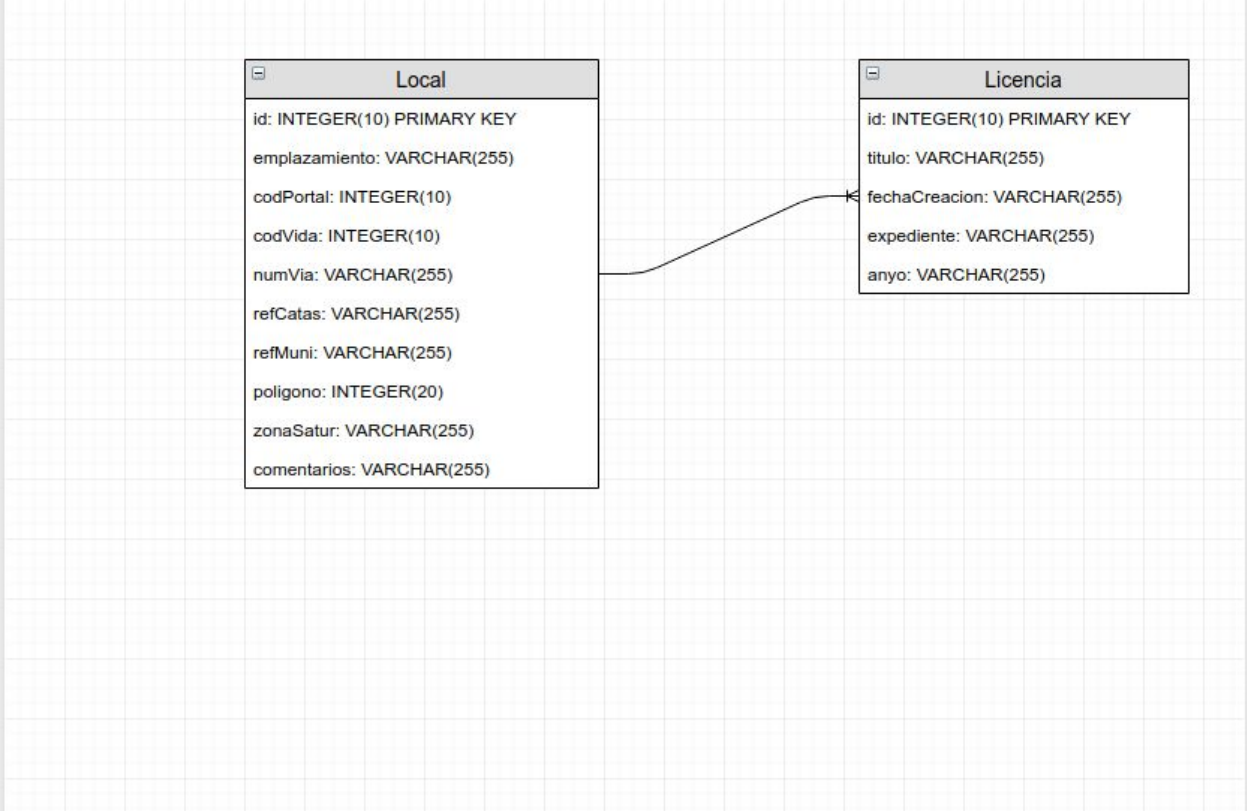
3.- Arbol de Nodos (DOM)



Debido a la complejidad del XML algunos nodos se ignoraron y solo se hizo enfasis en los utilizados en el proyecto. Si se desea se pueden consular todos los nodos en el XML proporcionado .

4.- JDBC

La estructura SQL, ENTIDAD - RELACION



El diagrama de entidad relacion para esta bbdd es sencilla, hay una relacion 1:M de Local con Licencia, por lo que Licencia guardará en su tabla un campo que referenciará a un id identificador de un registro de Local.

Que se va a guardar?

Los objetos a guardar en la base de datos serán los de tipo **Local** y **Licencia**. Con los datos porporcionados por el usuario o bien por un xml.

Script SQL (Resumido)

El script a continuación es una versión reducida del script completo, es reducido en el sentido de que algunos datos repetitivos fueron omitidos para no ocupar demasiado en el documento y verlo de una forma más simple. De todas formas se adjuntará un link donde se puede consultar el script en todo momento.

```sql
--
-- Índices para tablas volcadas
--

--
-- Indices de la tabla `licencia`
--
ALTER TABLE `licencia`
  ADD PRIMARY KEY (`Expediente`),
  ADD KEY `LocalID` (`LocalID`);

--
-- Indices de la tabla `local`
--
ALTER TABLE `local`
  ADD PRIMARY KEY (`ID`),
  ADD KEY `ID` (`ID`);

--
-- Restricciones para tablas volcadas
--

--
-- Filtros para la tabla `licencia`
--
ALTER TABLE `licencia`
  ADD CONSTRAINT `Licencia_ibfk_1` FOREIGN KEY (`LocalID`) REFERENCES `local` (`ID`) ON DELETE NO ACTION ON UPDATE NO ACTION;
COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

```sql
--
-- Estructura de tabla para la tabla `local`
--

CREATE TABLE `local` (
  `ID` int(10) NOT NULL,
  `Emplazamiento` varchar(255) NOT NULL,
  `CodigoPortal` varchar(255) NOT NULL,
  `CodigoVia` varchar(255) NOT NULL,
  `NumeroVia` varchar(255) NOT NULL,
  `RefCatas` varchar(255) NOT NULL,
  `RefMuni` varchar(255) NOT NULL,
  `Poligono` int(20) NOT NULL,
  `ZonaSaturada` varchar(255) NOT NULL,
  `Comentarios` varchar(255) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Volcado de datos para la tabla `local`
--

INSERT INTO `local` (`ID`, `Emplazamiento`, `CodigoPortal`, `CodigoVia`, `NumeroVia`, `RefCatas`, `RefMuni`, `Poligono`, `ZonaSaturada`, `Comentarios`) VALUES
(0, 'CALLE UNCETA, 92 ', '6898', '32440', '92', '4832516XM7143B', '74894,13208', 1, 'C', 'I 14/03/2018  EN EL LOCAL PONE QUE SE TRASPASA.'),
(1, 'CALLE MATIAS CARRICA, 11 (ZARAGOZA)', '861', '6220', '11', '6438608', '76527,13839', 1, 'C', 'ESQUINA CON C/ CONTAMINA'),
(2, 'CALLE MANIFESTACION, 2 (ZARAGOZA)', '869', '17980', '2', '6439402', '76464,13998', 1, 'C', '(Manifestacion,2)'),
(3, 'CALLE CIPRÉS, 4 (ZARAGOZA)', '855', '7520', '4', '6438414', '76424,13881', 1, 'C', 'Modificado horario según Acuerdo Municipal 23-05-06'),
(4, 'CALLE CONTAMINA, 2 (ZARAGOZA)', '863', '8220', '2', '6438601', '76484,13868', 1, 'C', 'Modificado horario según Acuerdo Municipal 23-05-06\nDECLARAR LA CADUCIDAD DE LA LICENCIA'),
(5, 'CALLE CONTAMINA, 3 (ZARAGOZA)', '903', '8220', '3', '6538309', '76488,13882', 1, 'C', 'Modificado horario según Acuerdo Municipal 23-05-06'),
(6, 'CALLE CONTAMINA, 5 (ZARAGOZA)', '904', '8220', '5', '6538307', '76508,13880', 1, 'C', 'Modificado horario según Acuerdo Municipal 23-05-06'),
(7, 'CALLE CONTAMINA, 6 (ZARAGOZA)', '1143', '8220', '6', '6438603', '76510,13858', 1, 'C', 'Modificado horario según Acuerdo Municipal 23-05-06'),
(8, 'CALLE CONTAMINA, 7 (ZARAGOZA)', '19633', '8220', '7', '6538307', '76508,13880', 1, 'C', 'Modificado horario según Acuerdo Municipal 23-05-06'),
```

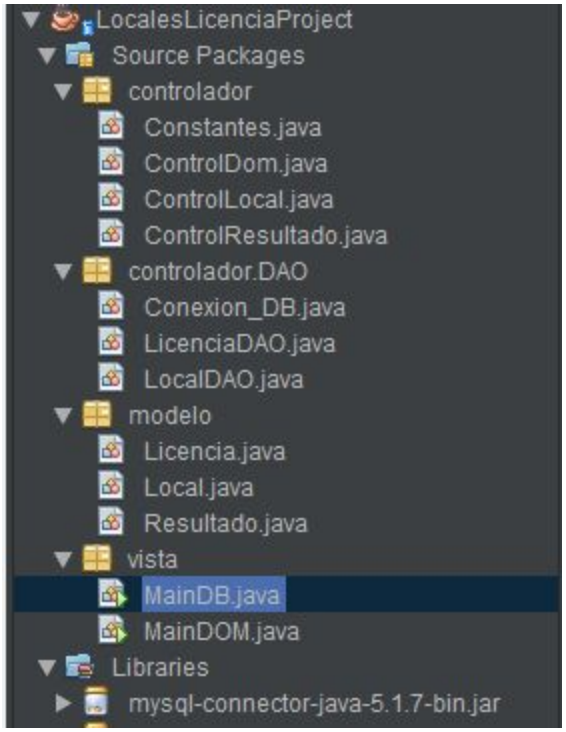
```sql
--
-- Estructura de tabla para la tabla `licencia`
--

CREATE TABLE `licencia` (
  `idLicencia` int(11) NOT NULL,
  `Titulo` varchar(255) NOT NULL,
  `FechaCreacion` varchar(255) NOT NULL,
  `Expediente` varchar(255) NOT NULL,
  `ANYO` varchar(255) NOT NULL,
  `LocalID` int(10) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Volcado de datos para la tabla `licencia`
--

INSERT INTO `licencia` (`idLicencia`, `Titulo`, `FechaCreacion`, `Expediente`, `ANYO`, `LocalID`) VALUES
(13, 'Licencia de Funcionamiento', '2007-05-25T00:00:00Z', '1013801', '2015', 26),
(29, 'COMUNICACIÓN PREVIA CAMBIO TITULARIDAD ACTIVIDAD', '2012-12-04T00:00:00Z', '1017275', '2014', 43),
(29, 'COMUNICACIÓN PREVIA CAMBIO TITULARIDAD ACTIVIDAD', '2012-12-04T00:00:00Z', '1025584', '2016', 14),
(1, 'Licencia de Apertura Actual', '2007-04-01T00:00:00Z', '1046690', '2005', 40),
(1, 'Licencia de Apertura Actual', '2007-04-01T00:00:00Z', '1063993', '2001', 25),
(16, 'Cambio de Titularidad de Licencia Funcionamiento', '2009-03-17T00:00:00Z', '1086634', '2010', 8),
(16, 'Cambio de Titularidad de Licencia Funcionamiento', '2009-03-17T00:00:00Z', '1103257', '2007', 8),
(29, 'COMUNICACIÓN PREVIA CAMBIO TITULARIDAD ACTIVIDAD', '2012-12-04T00:00:00Z', '1109796', '2015', 37),
(6, 'Licencia de Apertura', '2007-04-01T00:00:00Z', '112990', '2013', 4),
(24, 'Comunicación previa cambio Titularidad Apertura', '2012-02-28T00:00:00Z', '113318', '2012', 46),
(36, 'Declarar la caducidad', '2015-06-12T00:00:00Z', '1138566', '2014', 42),
(36, 'Declarar la caducidad', '2015-06-12T00:00:00Z', '1139109', '2014', 46),
(21, 'Comunicación Previa', '2011-09-30T00:00:00Z', '1139160', '2014', 40),
(13, 'Licencia de Funcionamiento', '2007-05-25T00:00:00Z', '1148810', '2011', 48),
```

Link archivo completo:

https://github.com/tihomir22/LocalesLicenciaProjectJAVAFINAL/blob/master/locales.sql

5.- Proyecto

La estructura del proyecto y sus paquetes:



El proyecto tiene 4 paquetes, aunque se podrian combinar en solo 3 , pero para mayor organización se crearon 2 paquetes diferentes para el controlador con el fin de separar el trabajo de DOM y la estructura de BBDD.

La estructura en detalle es:

- **Controlador**: Son todas las clases que van a ser usadas para obtener datos de los ficheros XML.
- **Controlador.DAO**: Son todas las clases que van a ser usadas para obtener datos de la base de datos y tambien la clase **Conexion_DB** será usada para establecer y cerrar la conexion a la base de datos.
- **Modelo**: Estas clases son los **POJO** aka “PIOJOS” , encargadas de dar una estructura a los objetos usados para distintos objetivos. Son los Plain Old Java Objects...
- **Vista:** Las clases con las que el usuario interactuará, son 2 para diferenciar la interacción con el XML y con la BBDD. Aunque se podrian combinar ambas en una GUI o web.

La estructura del proyecto y sus clases:

A continuación se explicará brevemente la función de cada clase del proyecto anteriormente mencionado.

1. **Constantes:** Es una clase con variables constantes, que permiten al programador acceder a distintas etiquetas del xml de forma más comoda.
2. **ControlDOM:** Clase usada para obtener el DOM a partir de ficheros XML y el proceso a la viceversa.
3. **ControlLocal:** Clase usada para devolver objetos Local a partir de un arbol de nodos DOM o añadirselos a uno vacio.
4. **ControlResultado:** Clase usada para deovlver objetos Resultado a partir de un arbol de nodos DOM o añadirselos a uno vacio.
5. **Conexion_DB:** Permitir el acceso a la base de datos y su respectivo control de estado.
6. **LicenciaDAO:** Un conjunto de metodos preparados explicitamente para la Clase Licencia con el fin de obtener datos, modificarlos, insertarlos o eliminarlos.
7. **LocalDAO:** Un conjunto de metodos preparados explicitamente para la Clase Local con el fin de obtener datos, modificarlos, insertarlos o eliminarlos.
8. **Licencia:** La estructura basica de Licencia usada para almacenar datos temporalmente.
9. **Local:** La estructura basica de Localusada para almacenar datos temporalmente.
10. **Resultado:** La estructura basica de Resultado usada para almacenar datos temporalmente.
11. **MainDB:** El usuario interactua con la BD.
12. **MainDOM:** El usuario interactua con los archivos XML y el DOM.

6.- Conclusiones

Este proyecto me ha servido muchisimo para aprender, sobretodo por el uso de un ejemplo de la vida real con un XML usado por un ayuntamiento real, de Zaragoza. Aunque la base de datos es simple, solo consta de dos tablas, lo compensa por su amplitud de datos y su magnitud a gran escala, donde un pequeño error puede corrompir toda la bbdd. Personalmente el proyecto me ha gustado aunque no la tematica, se podria mejorar para usar un tema más creativo y dinamico.

Estoy ansioso de seguir puliendo este proyecto o cualquiera que me venga encima!