MANUAL EXIST

22/12/2009

Manual per a desenvolupar una BD amb eXist

Autors:

David Prat Robles

david.prat-robles@est.fib.upc.edu

Toni Moreno Quiros

antonio.moreno@est.fib.upc.edu

Manual eXist

ÍNDEX

Manual eXist

INTRODUCCIÓ

Benvolgut lector, el manual que et presentem està basat en la nostra experiència en la creació d'una base de dades usant eXist i de la seva integració de dues maneres diferents dins d'un mateix projecte. Degut a la poca informació d'eXist i Xquery fins a la data, hem decidit fer aquest manual per ajudar als qui decideixin aventurar-se a usar aquesta tecnologia.

OBJECTIU DEL MANUAL

L'objectiu d'aquest manual és guiar al lector en el desenvolupament i integració d'una base de dades fent servir eXist.

QUÈ ÉS EXIST?

EXist és un sistema de gestió de bases de dades emmagatzemades en format XML, la qual cosa fa que segueixi un model jeràrquic.

PERQUÈ USAR EXIST?

El fet de que la informació estigui emmagatzemada en XML, fa que els resultats de les consultes es puguin conformar en XML. D'aquesta manera podem omplir una plana web amb el resultat d'una query directament.

Com a curiositat, cal saber que es podem emmagatzemar cadenes de caràcters molt llargs directament en els tags XML de les BDs; fet que queda restringit en SQL.

Per contra s'ha de tenir en compte que una BD jeràrquica és notablement més lenta que una relacional. És a dir, que no s'aconsella el seu ús amb volums de dades molt grossos on el temps de resposta sigui primordial.

INSTAL·LACIÓ D'EXIST

La instal·lació d'eXist és senzilla. Està totalment guiada en la següent direcció.

http://exist.sourceforge.net/quickstart.html

Cal a dir que la instal·lació és igual en Linux, en Windows i en Mac OS.

TREBALLANT AMB EXIST

En aquest manual mostrarem dues maneres de treballar amb eXist. Bàsicament, la primera s'usa en Java i la segona és apta per php.

XML-RPC + java

XML-RPC

XML-RPC és un protocol de crida a procediments remots (RPC) que fa servir XML per codificar les crides i HTTP com a mecanisme de transport de les dades.

Aquest protocol va anar evolucionant fins a convertir-se en el actual SOAP, que s'usa entre altres coses per a web services.

Estat de l'art

Actualment per treballar amb Java sobre eXist, el més comú és fer servir unes APIs per JAVA que fan servir XML-RPC per comunicar-se amb la BD; de manera que programar la interacció amb la BD és més simple, i pràcticament només haurem de formar les querys que voldrem fer sobre la BD.

Les APIs per JAVA

Sobre les APIs per Java, podeu trobar informació aquí:

http://exist.sourceforge.net/devguide_xmldb.html

Les podeu descarregar en la següent direcció:

http://www.filefront.com/15215763/undefined

El servidor no té límit de temps no obstant, si l'enllaç estigués caigut busqueu els següents fitxers:

exist.jar, jdom.jar, xmldb.jar, log4j-1.2.15.jar, xmlrpc-1.2.jar, jdom.jar.

No oblideu incloure les apis una per una en la variable d'entorn CLASSPATH.

Treballant amb Eclipse

És recomanable desenvolupar la nostra API en Java amb Eclipse, ja que existeixen dos pluguins molt útils que ens permetran agilitzar el desenvolupament.

El primer ens permet manegar les col·leccions de les BDs des del Eclipse i el segon ens permet fer querys directament sense haver de programar-les en la API de Java.

INSTAL·LANT ELS PLUGINS

- Ens posem d'amunt el menú "Help" i seleccionem "Software Updates".
- Seleccionem la pestanya "Avaliable Software"
- Fem click al botó "Add site".
- Posem la direcció: http://exist.sourceforge.net/plugin-update-site/.
- Fem click a "OK".
- Abans de cercar, marquem la casella dient que només cerqui les últimes versions dels plugins.
- Els seleccionem i els instal·lem.

HABILITANT ELS PLUGINS

- Anem al menú "Window".
- Seleccionem "Show view".
- Fem Click a "Other…".
- Seleccionem "eXist" i marquem les 3 vistes.

CONFIGURANT ELS PLUGUINS

- Anem a la pestanya "Browse View".
- Fem click a la icona situada a la dreta "Create new connection".
- Seleccionem "remote" i fem click a següent.
- Entrem la dirección xmldb:exist://xxx.xxx.xxx.xxx:8080/exist/xmlrpc on xxx.xxx.xxx.xxx será la IP de la màquina on estigui instal·lat el eXist. Si

està instal·lat a la mateixa màquina on estem desenvolupant ni haurà prou amb posar localhost.
- El nom de la connexió és indiferent.
- El username i el password han de correspondre's amb els que es tinguin al client d'eXist i que s'han de recordar de quan s'estava instal·lant eXist.
- Us apareixerà la nova connexió a la "Browse View".
- Feu click amb el botó dret i seleccioneu "connect".
- Teniu cura de tenir el eXist encès abans de intentar connectar.

Exemple de com fer la API de Java

A continuació es mostra un exemple complet de com fer una API en Java per manegar eXist.

Si opteu per fer un package com es fa al exemple, i que és el més normal, és possible que hagueu d'incloure en la variable d'entorn Path la ruta fins al directori on estiguin els executables del package.

També haureu d'incloure les APIS d'eXist en el projecte d'Eclipse.

Les següent línies de codi mostren tot el que cal per connectar-nos a la base de dades, així com la funció a la que passem un string amb la query que volem consultar la base de dades.

```
package datac;

import javax.xml.transform.OutputKeys;
import org.xmldb.api.DatabaseManager;
import org.xmldb.api.base.Collection;
import org.xmldb.api.base.Database;
import org.xmldb.api.base.Resource;
import org.xmldb.api.base.ResourceIterator;
import org.xmldb.api.base.ResourceSet;
import org.xmldb.api.base.XMLDBException;
import org.xmldb.api.modules.XPathQueryService;

public class DataController {

protected static String URI =
"xmldb:exist://localhost:8080/exist/xmlrpc/dbMusic";

public static Collection col;
public static XPathQueryService service;

      //Connecta a al BD
      public void connect() throws Exception {
            String driver = "org.exist.xmldb.DatabaseImpl";
            // initialize database driver
            Class<?> cl = null;
            try {
            cl = Class.forName(driver);
            } catch (Exception e) {
            throw new Exception("Error en inicialitzacio del driver1");
```

```
            }
            Database database = null;
            try {
            database = (Database) cl.newInstance();
            } catch (InstantiationException e) {
            throw new Exception("Error en inicialitzacio del driver2");
            } catch (IllegalAccessException e) {
            throw new Exception("Error en inicialitzacio del driver3");
            }
            try {
            DatabaseManager.registerDatabase(database);
            database.setProperty("create-database", "true");
            } catch (XMLDBException e) {
            throw new Exception("Error en inicialitzacio del driver4");
            }
            try {
            col = DatabaseManager.getCollection(URI,"admin","admin");
            col.setProperty(OutputKeys.INDENT, "no");
            service = (XPathQueryService)
            col.getService("XPathQueryService","1.0");
            service.setProperty("pretty", "true");
            service.setProperty("encoding", "8859-1");
            service.setProperty("indent", "yes");

            } catch (XMLDBException e) {
            throw new Exception("Error en XMLDB");
            }
      }

      //Fa una query del tipus que sigui a la BD
      private String doQuery(String query) throws Exception {
            String res = "";
            try {
            ResourceSet result = service.query(query);

            ResourceIterator i;
            i = result.getIterator();
            while (i.hasMoreResources()) {
                  Resource r = i.nextResource();

                  res = res.concat((String) r.getContent());
                  }
            } catch (Exception e) {
            throw new Exception("Query Error");
            }
            return res;
      }

}
```

Haureu de fer un nou “java project” amb Eclipse, englobar-hi la classe DataController i Empaquetador, que veurem a la següent secció. El fitxer per fer les probes es recomanable definir-lo fora del package perquè no formarà part del sistema final. Les funcions que contindran les querys, que veurem més endavant; s’han d’inserir abans de l’ultima clau del codi del DataController.

Usant JDOM

Com ja hem comentat la manera ideal per treballar amb eXist, es retornant els resultats de les consultes directament en XML. No obstant, és molt possible que hagueu de passar els resultats en un classe de Java o hagueu d'extreure només una part de la informació que retorna la consulta.

Per això, en aquest manual també tractarem exemples de com fer-ho.

```
package datac;

import java.io.StringReader;
import java.util.Iterator;
import java.util.List;
import java.util.Vector;

import org.jdom.Document;
import org.jdom.Element;
import org.jdom.input.SAXBuilder;
import org.jdom.output.Format;
import org.jdom.output.XMLOutputter;
import org.xml.sax.InputSource;


public class Empaquetador {

	public class csong{
		public String artist;
		public String album;
		public String track;
		public String genre;
		public String year;
	}

	public class cpersona{
		public String username;
		public String password;
		public String mail;
		public String numplaylists;
		public String esautor;
	}

	//ens retorna un document Document a partir d'un resultat d'una query
	//que lògicament hauria de tenir fomat XML
	public Document obteDocument (String Query) throws Exception{
	    SAXBuilder sa = new SAXBuilder();
	    StringReader sr = new StringReader(Query);
	    InputSource is = new InputSource(sr);
	    Document doc = sa.build(is);

	    return doc;
	}

	//Esciur pel canal estàndard de sortida el contingut del Document
	public  void outputDocument(Document myDocument) {
	    try {
	      XMLOutputter outputter = new
	      XMLOutputter(Format.getPrettyFormat());
	      outputter.output(myDocument, System.out);
```

```
    } catch (java.io.IOException e) {
      e.printStackTrace();
    }
}

//obté un element d'un document
public  String obteElement(Document doc, String XMLTag){
    return doc.getRootElement().getChildText(XMLTag);
}

//obté varis element d'un document i els emmagatzema en un Vector
//està pensada només per a la query que retorna un conjunt de cançons
public  Vector obteElements(Document doc){

    Vector vsongs = new Vector();

    Element e3=doc.getRootElement();
    List ms=e3.getChildren();
    Iterator itr=ms.iterator();

    while(itr.hasNext()){
      Element me= (Element) itr.next();
      Element me2= me.getChild("artist");
      Element me3= me.getChild("album");
      Element me4= me.getChild("track");
      Element me5= me.getChild("genre");
      Element me6= me.getChild("year");

      csong ma = new csong();
      ma.artist=me2.getText();
      ma.album=me3.getText();
      ma.track=me4.getText();
      ma.genre=me5.getText();
      ma.year=me6.getText();

      vsongs.add(ma);
    }
    return vsongs;
}

//obté varis element d'un document i els emmagatzema en un Vector
//està pensada per obtenir el nom de les llistes que hi ha a doc
public Vector obtellistes(Document doc){

      Vector vlistes = new Vector();
      Element e3=doc.getRootElement();
      List ms=e3.getChildren();
      Iterator itr=ms.iterator();


    while(itr.hasNext()){
      Element me= (Element) itr.next();
      Element me2= me.getChild("nomllista");
      String s=me2.getText();

      vlistes.add(s);
    }
      return vlistes;
}
```

```
	//obté els element d'un document on hi ha el resultat de cercar
	//la informació d'una persona, concretament la funció consultaruser.
	public  cpersona obteElementspersona(Document doc){

	    Element e3=doc.getRootElement();
	    cpersona cper=new cpersona();

	    Element me2= e3.getChild("username");
	    Element me3= e3.getChild("password");
	    Element me4= e3.getChild("mail");
	    Element me5= e3.getChild("numplaylists");
	    Element me6= e3.getChild("esautor");

	    cper.username=me2.getText();
	    cper.password=me3.getText();
	    cper.mail=me4.getText();
	    cper.numplaylists=me5.getText();
	    cper.esautor=me6.getText();

	      return cper;
	}

	//obté els elements d'una canço que ha esta els resultat d'una cerca
	//emmagatzemada a al Document doc
	public csong obteElementssong(Document doc){
	      Element e3=doc.getRootElement();
	    csong cso=new csong();

	    Element me2= e3.getChild("artist");
	    Element me3= e3.getChild("album");
	    Element me4= e3.getChild("track");
	    Element me5= e3.getChild("genre");
	    Element me6= e3.getChild("year");

	    cso.artist=me2.getText();
	    cso.album=me3.getText();
	    cso.track=me4.getText();
	    cso.genre=me5.getText();
	    cso.year=me6.getText();

	      return cso;
	}

}
```

Curs de querys amb Xquery

Aquest apartat del manual possiblement us tregui moltes hores de feina. Ja que la manera de pensar per fer querys en una base de dades jeràrquica és diferent que fer-les en una base de dades relacional. Per fer les querys en Xquery hem de pensar en arbres XML i com recorre'ls amb punters, en comptes de pensar en taules i com seleccionar-ne les files i columnes que volem.

Per entendre les querys que mostrarem es necessita saber la estructura de la base de dades.

La següent base de dades vol gestionar les dades del que vindria a ser un Spotify o un LastFM; es a dir, un servei de música per streaming.

El primer fitxer guarda la informació de les persones, aquest fitxer estarà a la col·lecció DBMusic i s'anomenarà dataPersones.

El podem crear mitjançant el client d'eXist o amb el plugin de l'Eclipse.

Per crear-lo amb el plugin d'Eclipse, farem el següent:

- A la "Browse view" fem click amb el botó dret del ratolí a la connexió que volem.
- Seleccionem "Open".
- Entrem dins del directori "db".
- Creem la nova collection amb el nom "DBMusic".
- Fem click amb el botò dret a DBMusic y seleccionem "create document"
- L'anomenem dataPersones i Enganchem el contingut de sota

```
<dataPersones>
    <persones>
        <persona>
            <username>Donald</username>
            <password>123</password>
            <mail>donald@gmail.com</mail>
            <numplaylists>0</numplaylists>
            <esautor>si</esautor>
        </persona>
        <persona>
            <username>Pere</username>
            <password>123</password>
            <mail>pere@gmail.com</mail>
            <numplaylists>0</numplaylists>
            <esautor>no</esautor>
        </persona>
    </persones>
</dataPersones>
```

Procedim de manera igual pel fitxer que guarda les cançons però, òbviament l'anomenem dataSongs.

```
<dataSongs>
    <songs>
        <song>
            <artist>Steely Dan</artist>
            <album>Aja</album>
            <track>Aja</track>
            <file>./steelydan/aja/aja.mp3</file>
            <genre>Jazz-Rock</genre>
            <year>1992</year>
            <userid>Donald</userid>
```

```
        </song>
        <song>
            <artist>Steely Dan</artist>
            <album>Aja</album>
            <track>Blac Cow</track>
            <file>./steelydan/aja/blackcow.mp3</file>
            <genre>Jazz-Rock</genre>
            <year>1992</year>
            <userid>Donald</userid>
        </song>
        <song>
            <artist>Steely Dan </artist>
            <album>Countdown to Ecstasy</album>
            <track>Bodhisattva</track>
            <file>./steelydan/countdowntoecstasy/bodhisattva.mp3</file>
            <genre>Jazz-Rock</genre>
            <year>1973</year>
            <userid>Donald</userid>
        </song>
        <song>
            <artist>AC/DC</artist>
            <album>Razors Edge</album>
            <track>Thunderstruck</track>
            <file>./acdc/razorsedge/thunderstruc.mp3</file>
            <genre>Rock</genre>
            <year>1991</year>
            <userid>Donald</userid>
        </song>
    </songs>
</dataSongs>
```

Finalment el fitxer que conté les llistes de reproducció. És el més interessant per aprendre a fer querys degut a la seva estructura.

```
<dataLlistes>
    <llistes>
        <llista>
            <username>Donald</username>
            <nomllista>rocklist</nomllista>
            <songref>
                <album>Aja</album>
                <track>Aja</track>
            </songref>
            <songref>
                <album>Aja</album>
                <track>Black Cow</track>
            </songref>
        </llista>
        <llista>
            <username>Donald</username>
            <nomllista>rocklistdonald</nomllista>
            <songref>
                <album>Aja</album>
                <track>Aja</track>
            </songref>
```

```
            <songref>
                <album>Aja</album>
                <track>Black Cow</track>
            </songref>
            <songref>
                <album>Countdown to Ecstasy</album>
                <track>Bodhisattva</track>
            </songref>
        </llista>
        <llista>
            <username>Pere</username>
            <nomllista>mylist1</nomllista>
            <songref>
                <album>Aja</album>
                <track>Aja</track>
            </songref>
            <songref>
                <album>Aja</album>
                <track>Black cow</track>
            </songref>
        </llista>
        <llista>
            <username>Pere</username>
            <nomllista>mylist2</nomllista>
            <songref>
                <album>Razors Edge</album>
                <track>Thunderstruck</track>
            </songref>
        </llista>
        <llista>
            <username>Pere</username>
            <nomllista>myemptylist</nomllista>
        </llista>
    </llistes>
</dataLlistes>


//ANOTACIONS BD//
/*el userid de dataSongs es el username de dataPersones
* dataPersones primary key username foreign key ""
* dataSongs primary key album+track foreign key useid references username in
* dataPersones
* dataLlistes primary key username+nomllista foreign key album+track references
* a song in dataSongs
* , restriccio només hi pot haver una canso amb album+track dins d'una mateixa
* llista.
*/
```

Comencem fent algunes consultes bàsiques:

La següent consulta cerca en el fitxer dataPersones i navega en l'arbre XML fins al tag persona. Entre corxets especifiquem que volem només la persona que té com a username amb el paràmetre usern. Que de fet, és la nostra clau primària per a una persona.

Pel que fa les claus primàries les haurem de controlar nosaltres mateixos al inserir una nova persona, (i per tots els casos). És a dir, que al inserir una persona abans

haurem de fer una query preguntant si ja existeix una persona amb el mateix nom usern.

És necessari posar entre cometes dobles el paràmetre, per a fer-ho en Java hem d'escapar el caràcter " amb \".

La part /username/text(), és per indicar que volem que ens retorni en el string result només el que conté el tag username de la persona. /text(), concretament, és per dir que no ens retorni els tags que envolten el contingut. Això vol dir que sinó, poséssim el /text() ens retornaria el username envoltat pels tags.

```
public String existeixuser(String usern) throws Exception{
	String result ="";
	result=doQuery("/dataPersones/persones/persona[username =
	\""+usern+"\"]/username/text()");
	return result;
}
```

Per fer la mateixa query sense haver de córrer el projecte Java es pot fer servir el plugin d'Eclipse. Es procedeix de la següent manera:

- Fem click amb el botó dret del ratolí a DBMusic.
- Seleccionem "run query (deprecated)".
- Inserim la següent query:
- /dataPersones/persones/persona[username = "Donald"]/username/text().
- Obtenim Donald.
- Ara proveu de fer la mateixa query sense /text().

Com fer una inserció?

```
public int insereixUsuari(String usern, String pass, String mail, String
numplaylists, String esautor) throws Exception{

doQuery("update insert
<persona><username>"+usern+"</username><password>"+pass+"</password>" +
"<mail>"+mail+"</mail><numplaylists>"+numplaylists+"</numplaylists>" +
"<esautor>"+esautor+"</esautor></persona>into /dataPersones/persones");

return 0;
}
```

Com ja hem dit la comprovació de que no existeix ja un usuari amb el mateix username s'hauria de fer abans; fent una query. Això és millor fer-ho en la mateixa funció insereixUsuari, ja que facilita l'ús de la vostra API.

Com fer una modificació?

```
public void modificaNumplaylists(String username, String numplaylists) throws
Exception{
	doQuery("update replace /dataPersones/persones/persona[username =
	\""+username+"\"]/numplaylists/text() with  \""+numplaylists+"\"");
}
```

Com fer una eliminació?

```
public void esborrarUsuari(String usern) throws Exception{
	doQuery("update delete /dataPersones/persones/persona[username =
	\""+usern+"\"]");
}
```

Anem a veure ara algunes querys més complexes que ús seran molt útils. Ja que per fer joins, cerques per varis possibles valors, etc. Al contrari que SQL, per a Xquery no hi ha informació d'aquesta complexitat enlloc, i vaig haver d'inventar-me com fer-ho, (com a bon "Fiber" ;-)).

Degut al límit en la extensió del manual es comentaran breument, tanmateix recomano que sapigueu que fan, en llegiu i compreneu la seva sintaxi i en comproveu el seu resultat.

Exemple 1: Retornant varis camps d'un node.

```
//ens retorna la informació associada a un usuari
public String consultauser(String usern) throws Exception{
	String result ="";

	result=doQuery("let $persona:=/dataPersones/persones/persona[username =
	\""+usern+"\"]" +
	"return <persona>" +
	"<username>{$persona/username/text()}</username>" +
	"<password>{$persona/password/text()}</password>" +
	"<mail>{$persona/mail/text()}</mail>" +
	"<numplaylists>{$persona/numplaylists/text()}</numplaylists>" +
	"<esautor>{$persona/esautor/text()}</esautor>" +
	"</persona>");

	return result;
}
```

Exemple 2: cerca segons varis possibles valors.

```
//query multivaluda segons varis posibles paràmetres
public  String composedquery(String artist, String album, String track, String
genre, String year)throws Exception{
		String result="";
		//artist
		if (artist==""){
			artist="not(empty($song/artist))";
			//System.out.println("file");
		}else{
			artist="$song/artist = \""+artist+"\"";
		}//album
		if (album==""){
			album="not(empty($song/album))";
			System.out.println("album:"+album);
		}else{
			album="$song/album = \""+album+"\"";
		}//track
		if (track==""){
			track="not(empty($song/track))";
		}else{
```

```
                track="$song/track = \""+track+"\"";
                System.out.println("track:"+track);
            }//genre
            if(genre==""){
                genre="not(empty($song/genre))";
            }else{
                genre="$song/genre = \""+genre+"\"";
            }//year
            if(year==""){
                year="not(empty($song/year))";
            }else{
                year="$song/year = \""+year+"\"";
            }

            result=doQuery("<songs>{" +
                        "let $songs:=/dataSongs/songs/song " +
                        "for $song in $songs where "+artist+ " and "+album+" and
                        "+track+" and "+genre+" and "+year+" " +//important que
                        hi hagi espai entre songsr i " sino no funciona
                        "return <song>" +
                        "<artist>{$song/artist/text()}</artist>" +
                        "<album>{$song/album/text()}</album>" +
                        "<track>{$song/track/text()}</track>" +
                        "<genre>{$song/genre/text()}</genre>" +
                        "<year>{$song/year/text()}</year>" +
                        "</song>}" +
                        "</songs>");
        return result;
}
```

Exemple 3: cercant a varis documents, (Join).

Com veieu es tracta de definir punters per als arbres XML, amb la sentència "let", i usar-los per obtenir informació segons ens convingui.

```
//retorna totes les cançons que té la llista que té el user i nom usern i
nomllista
public String consultarsongsllista (String usern, String nomllista) throws
Exception{
        String result="";

        result=doQuery("<songs>{" +
        "let $list:=/dataLlistes/llistes/llista[username= \""+usern+"\" and
        nomllista= \""+nomllista+"\"]" +
        "let $songref:=$list/songref " +
        "for $song in $songref " +
        "let $songcolle:=/dataSongs/songs/song[album= $song/album/text() and
        track=$song/track/text()]" +
        "return <song>" +
        "<artist>{$songcolle/artist/text()}</artist>" +
        "<album>{$list/$song/album/text()}</album>" +
        "<track>{$list/$song/track/text()}</track>" +
        "<genre>{$songcolle/genre/text()}</genre>" +
        "<year>{$songcolle/year/text()}</year>" +
        "</song>}" +
        "</songs>");

        return result;
}
```

Exemple 4: Quan es necessiten dos punters per recórrer un arbre XML en Xquery.

Aquest és un cas peculiar de Xquery, ja que quan apuntem fins a algun nivell de l'arbre XML perdem la visibilitat dels nodes superiors respecte al punter.

```
//comproba si una canço está en una llista
public  String comprobaSongaLlista(String usern, String nomllista, String album,
String track) throws Exception{
	String result="";

	result=doQuery("<tracks>{" +
	"let $lists:=/dataLlistes/llistes/llista[username= \""+usern+"\" and
	nomllista= \""+nomllista+"\"]" +
	"let $songref:=/dataLlistes/llistes/llista/songref[album= \""+album+"\"
	and track= \""+track+"\"]" +
	"for $list in $lists where songref/album = \""+album+"\" and songref/track
	= \""+track+"\" " + //important que hi hagi espai entre songsr i " sino no
	funciona
	"return <track>" +
	"<album>{$list/$songref/album/text()}</album>" +
	"<track>{$list/$songref/track/text()}</track>" +
	"</track>}" +
	"</tracks>");

	return result;
}
```

PHP + SOAP

SOAP

SOAP en exist apareix com una interfície d'accés alternativa al XML-RPC, que ens permet accedir des de qualsevol llenguatge de programació que doni suport a aquest servei.

En el nostre cas l'utilitzarem per accedir des de PHP, per el que és important que el mòdul SOAP estigui introduït a la versió de PHP que estiguem utilitzant. En cas contrari, es necessitarà instal·lar-lo. Normalment en els paquets de php5 dels repositoris estàndards que tenen la majoria de versions de Linux, ja tenen aquest servei integrat.

Quins serveis SOAP ofereix exist?

EXist utilitza el paquet d'eines Axis SOAP Apache que corre com un servlet. Aquest està configurat automàticament escoltant el port 8080, suportant els serveis establerts a http://localhost:8080/exist/services . D'aquesta forma podem accedir al SOAP, però sempre dependents del servidor.

Els serveis que posa eXist a la nostra disposició, divideixen el tracte amb la bases de dades en operacions de consulta i d'administració. Possiblement existeixen més serveis, però aquest son els utilitzats per nosaltres:

- Operacions de **consulta**:
  http://localhost:8080/exist/services/Query?wsdl .
- Operacions d'**administració:**
  http://127.0.0.1:8080/exist/services/Admin?wsdl

Per comprovar que els serveis estan accessibles, recomano copiar les urls exposades al navegador, tenint instal·lat i corrent l'exist. D'aquesta forma es mostrarà el codi que pertany al servei corresponent.

Exemple del servei de les operacions de consulta:

```
– <wsdl:definitions targetNamespace="urn:exist">
  – <!--
     WSDL created by Apache Axis version: 1.4
     Built on Apr 22, 2006 (06:55:48 PDT)
    -->
  – <wsdl:types>
    – <schema elementFormDefault="qualified" targetNamespace="urn:exist">
      – <element name="getResource">
        – <complexType>
          – <sequence>
              <element name="sessionId" type="xsd:string"/>
              <element name="path" type="xsd:string"/>
              <element name="indent" type="xsd:boolean"/>
              <element name="xinclude" type="xsd:boolean"/>
            </sequence>
          </complexType>
        </element>
      – <element name="getResourceResponse">
        – <complexType>
          – <sequence>
              <element name="getResourceReturn" type="xsd:string"/>
            </sequence>
          </complexType>
        </element>
      – <element name="query">
        – <complexType>
          – <sequence>
              <element name="sessionId" type="xsd:string"/>
              <element name="xpath" type="xsd:string"/>
            </sequence>
          </complexType>
        </element>
                                                                    ...
```

Però no serà necessari l'esforç per entendre i identificar les funcions que s'ofereixen als serveis, ja que contactarem amb ells mitjançant una llibreria: **phexist**.

PHEXIST

És una llibreria lliure feta per Oscar Celma i Xavier Prunayre que facilita la crida de les funcions del servei per realitzar les consultes i manipular les col·leccions.

Tant per descarar-la com per obtenir més informació podeu visitar:

http://query-exist.sourceforge.net

Analitzant la llibreria PHEXIST

Primerament trobem que la llibreria es divideix en dos classes, diferenciant quan treballen amb els dos serveis descrits:

- Class eXist
- Class eXistAdmin extends eXist

La classe que dona servei a l'administració es va realitzar posteriorment a l'altre, per el que va aprofitar la seva estructura mitjançant l'herència.

Class eXist: Anàlisis

Les funcions que proporcionen, amb una petita descripció, són les següents:

- **__construct**($user="guest",$password="guest",$wsdl="http://localhost:8080/exist/services/Query?wsdl")
  La constructora amb els parametres necesaris per la conexió, l'user i password de la bases de dades, i el servei de consulta. A més per defecte ja venen aquest possats.

- **__destruct**()
  Que defineix la destrucció per defecte.

- **Connect**()
  Per conectar-se a la base de dades facilitada a la constructora.

- **Disconnect**()

Per desconectar-se.

- **getError**()
  Per obtenir els posibles errors generats al utilitzar el servei.

- **Xquery**($query)
  Per realitzar la consulta query. Les analitzarem en l'apartat següent.

- **setHighlight**($highlight), **setDebug**($debug=true), **setUser**($user), **setPassword**($passwd), **setWSDL**($wsdl), **setError**($error).
  Per introduir les dades directament. Aquestes funcions no ens han calgut utilitzarles.

Class eXist : Utilització

Les querys es realitzen de la mateixa forma que les de consulta de Xquery. Així que podrem re aprofitar les mateixes que les realitzades en una api Java, tenint en compte que el llenguatge que estem utilitzant és PHP. Així que hem de tenir molta cura amb els caràcters reservats escampant-los quan sigui necessari. A continuació mostro unes quantes consultes que es realitzen sobre les mateixes taules que les abans mostrades.

- $query = '/dataPersones/persones/persona[username = "'.$username.'" and password = "'.$password.'"]';
  Obté la persona amb l'username i password especificat.

- $query = "/dataLlistes/llistes/llista[username= '".$username."' and nomllista= '".$nomllista."']/username/text()";
  Obté una llista donat un username i un nomllista

- $query = "<llistes>{let \$llistes:=/dataLlistes/llistes/llista[username= '".$username."']for \$llista in \$llistes return <llista><nomllista>{\$llista/nomllista/text()}</nomllista><username>{\$llista/username/text()}</username></llista>}</llistes>";
  Obté totes les llistes d'un usuario especificat username. En aquest cas cal destacar l'escapament dels **$** que no fan referència a les variables de php.

- $query = "<songs>{let \$songs:=/dataSongs/songs/song[".$var1." = '".$busqueda."']for \$song in \$songs return

```
<song><artist>{\$song/artist/text()}</artist><album>{\$song/album/tex
t()}</album><track>{\$song/track/text()}</track><genre>{\$song/genr
e/text()}</genre><year>{\$song/year/text()}</year></song>}</songs>
";
```

Obté els songs on l'etiqueta especificada en $var1, pren el valor contingut en $busqueda. Aquesta consulta s'ha feta aprofitant el potencial de concatenament de text de php.

Al realitzar les consultes el resultat ens el retornen en un XML. Aquest es pot aprofitar directament com vam comentar, o bé accedir directament al contingut que ens interessa.

En el cas de voler treballar directament amb els continguts de les etiquetes com si es tractes de qualsevol altre bases de dades, podem aprofitar-nos del sistema de vectors de PHP. Per aconseguir-lo directament i d'una forma fàcil, hem empleat una altre llibreria lliure anomenada **xmltoarray**, que ens proporciona un pas directe de xml a un array de php.

El xmltoarray funciona de forma molt simple. N'hi ha prou amb crear un objecte del tipus XMlToArray passant-li la variable xml, i després cridar la funció createArray(). Un exemple senzill seria el següent:

```
//obtenim el xml

$res = $result["XML"];

//creem una instancia de la clase amb el xml

$xmlObj    = new XmlToArray($res);

//obtenim l'array

$arrayData = $xmlObj->createArray();
```

D'aquesta forma tenim en un array $arrayData el contingut del xml, on podem accedir al contingut de l'etiqueta "etiqueta" de la forma $arrayData[`etiqueta']. En el cas que la consulta retornés varies instancies, tindríem cadascuna de les instancies en la posició "i", sent "i" un nombre entre 0 i el numero d'instàncies. Cal destacar aquest aprofitament del potencial de l'array de PHP.

Class eXistAdmin : Anàlisis

A més de les funcions de la classe exists, ja que hereten d'aquesta, la classe eXistAdmin proporciona:

- **__construct**($user="guest",$password="guest",$wsdl="http://localhost:8080/exist/services/Admin?wsdl")
  La contructora, de la mateixa forma que a la classe exist, però que ara en lloc de tenir el servei de query per defecte, té el de admin.

- **store**($data, $encoding = "UTF-8", $path = "/db", $replace = false)
  Es pot utilitzar per crear noves taules especificada a $data, creant-la al $path especificat, amb el format de text $encoding, que reemplaçarà si existeix en el cas que $replace tingui el valor de cert. Aquesta funció no la utilitzem al projecte, tot i que la vam trobar molt interessant.

- **createCollection**($path),**removeCollection**($path),**removeDocument**($path).
  Per crear col·leccions, eliminar col·leccions i eliminar taules respectivament. Només cal especificar-li el path.

- **xupdateResource**($documentName, $xupdate)
  La funció amb la que realitzarem les crides a la base de dades, passant-li el path de la taula de la forma : '/db/dbMusic/dataSongs' per exemple; i la Xupdate que explicarem la seva realització en la utilització.

Class eXistAdmin: Utilització

Per realitzar l'administració de la bases de dades, construirem les crides utilitzant Xupdate. Més concretament ens bastarà amb les seves funcionalitats de:

- **xupdate:update**. Per modificar instancies ja ficades a la base de dades.

- **xupdate:remove**. Per eliminar instancies ja ficades a la base de dades.

- **xupdate:append**. Per introduir noves instancies a la base de dades. Concretament ens crearem els nous elements amb **xupdate:element** .

Es pot trobar informació d'aquesta utilització i de totes les funcionalitats del xupdate a la web http://xmldb-org.sourceforge.net/xupdate/xupdate-wd.html . Recomano la seva visita, tot i que n'hi ha coses com l'accés per etiqueta especifica, que no vam trobar explicada. Tot seguit en els nostres exemples si que trobareu com realitzar això.

A continuació analitzarem mitjançant un exemple de cada, aquestes tres funcionalitats:

- **xupdate:append + xupdate:element**.

```
$xupdate = '<?xml version="1.0"?><xupdate:modifications version="1.0" xmlns:xupdate="http://www.xmldb.org/xupdate">
 <xupdate:append select="/dataLlistes/llistes/llista[username=\''.$username.'\' and nomllista=\''.$mylista.'\']" child="last()">
   <xupdate:element name="songref">
       <album>'.$album.'</album>
    <track>'.$titulo.'</track>
   </xupdate:element>
 </xupdate:append>
</xupdate:modifications>';
```

Mitjançant aquesta Xuptade, fem un insert en la llista amb username = $username, i nomllista = $mylista . Per poder fer l'insert d'aquest nou element, el creem mitjançant Xupdate:element amb dos tag. Un tag es <album> amb el contingut de la variable $album; i l'altre es <track> amb el contingut de la variable $titulo.

- **xupdate:remove**.

```
$xupdate = '<?xml version="1.0"?><xupdate:modifications version="1.0" xmlns:xupdate="http://www.xmldb.org/xupdate">
<xupdate:remove select="/dataLlistes/llistes/llista/songref[album=\''.$album.'\' and track=\''.$title.'\']">
</xupdate:remove>
</xupdate:modifications>' ;
```

Amb aquesta Xupdate, utilitzant el remove, eliminem el songref amb tag album el contingut de la variable $album, i tag $track el contingut de la variable $title.

- **xupdate:update.**

```
$xupdate = '<xupdate:modifications version="1.0" xmlns:xupdate="http://www.xmldb.org/xupdate">
<xupdate:update select="/dataPersones/persones/persona[username=\''.$usuario.'\']/password">'.$password.'</xupdate:update>
</xupdate:modifications>';
```

Mitjançant aquesta Xupdate, cambiem el contigut del tag password de la persona amb username = $usuario, pel contingut de la cariable $password.

Creació de les APIS PHP

Després d'haver llegit les explicacions de les queris de consulta i administració, amb la utilització de phexist i xmltoarray, suposo que ja sabríeu realitzar les apis per tractar les bases de dades.

Per últim puc recomanar la utilització d'una api per cada taula, per aconseguir una millor estructuració, a més de si fos necessari, poder utilitzar patrons passarel·la guardant sempre una instancia en la api.

Com exemple posaré una de les apis empleades en aquest projecte, la que tracta la taula Songs, composada tal i com es mostra a la pàgina 10 – 11.

```php
<?php
include_once ('include/eXist.php');
require_once "./include/class.xmltoarray.php";

class dataSongs{
    public function __construct()()

    public function filtrarLista($busqueda, $filtro)
    {
        $db = new eXist("admin", "admin");
        $db->connect();
        if (strcmp($filtro,'todos')==0)
        {
            $query = "<songs>{let \$songs:=/dataSongs/songs/song for \$song in \$songs return <song>
            <album>{\$song/album/text()}</album><track>{\$song/track/text()}</track></song>}</songs>";
        }
        else
        {
            if (strcmp($filtro, 'titulo')==0) $var1= 'track';
            if (strcmp($filtro, 'album')==0) $var1= 'album';
            if (strcmp($filtro, 'artista')==0) $var1= 'artist';

            $query = "<songs>{let \$songs:=/dataSongs/songs/song[".$var1." = '".$busqueda."']for \$song in \$songs
            return <song><artist>{\$song/artist/text()}</artist><album>{\$song/album/text()}</album>
            <track>{\$song/track/text()}</track><genre>{\$song/genre/text()}</genre><year>{\$song/year/text()}</year></song>}</songs>";
        }
        $db->setHighlight(FALSE);
        $result = $db->xquery($query);
        if ( !empty($result["XML"]) )
        {
            $res = $result["XML"];
            //Creating Instance of the Class
            $xmlObj    = new XmlToArray($res);
            //Creating Array
            $arrayData = $xmlObj->createArray();
        }
        $db->disconnect();

        return $arrayData['songs']['song'];
    }
```

```php
public function introducirCancion($artist, $album, $title, $file, $genre, $year,  $userid)
{
    $db = new eXistAdmin('admin', 'admin', 'http://127.0.0.1:8080/exist/services/Admin?wsdl');
    $db->connect() or die($db->getError());

    $artist = stripcslashes($artist);
    $album = stripcslashes($album);
    $title = stripcslashes($title);
    $genre = stripcslashes($genre);
    $year = stripcslashes($year);
    $userid = stripcslashes($userid);

    $xupdate = '<?xml version="1.0"?><xupdate:modifications version="1.0" xmlns:xupdate="http://www.xmldb.org/xupdate">
     <xupdate:append select="/dataSongs/songs" child="last()">
       <xupdate:element name="song">
           <artist>'.$artist.'</artist>
        <album>'.$album.'</album>
        <track>'.$title.'</track>
        <file>'.$file.'</file>
        <genre>'.$genre.'</genre>
        <year>'.$year.'</year>
        <userid>'.$userid.'</userid>
       </xupdate:element>
     </xupdate:append>
    </xupdate:modifications>';
    $db->xupdateResource('/db/dbMusic/dataSongs', $xupdate);
    $db->disconnect();
}
```

```php
public function eliminarCancion($album, $title)
{

    $db = new eXistAdmin('admin', 'admin', 'http://127.0.0.1:8080/exist/services/Admin?wsdl');
    $db->connect() or die($db->getError());
    $album = stripcslashes($album);
    $title = stripcslashes($title);
    $xupdate = '<?xml version="1.0"?><xupdate:modifications version="1.0" xmlns:xupdate="http://www.xmldb.org/xupdate">
    <xupdate:remove select="/dataSongs/songs/song[album=\''.$album.'\' and track=\''.$title.'\']">
    </xupdate:remove>
    </xupdate:modifications>';
    $db->xupdateResource('/db/dbMusic/dataSongs', $xupdate);
    $xupdate = '<?xml version="1.0"?><xupdate:modifications version="1.0" xmlns:xupdate="http://www.xmldb.org/xupdate">
    <xupdate:remove select="/dataLlistes/llistes/llista/songref[album=\''.$album.'\' and track=\''.$title.'\']">
    </xupdate:remove>
    </xupdate:modifications>' ;
    $db->xupdateResource('/db/dbMusic/dataLlistes', $xupdate);
    $db->disconnect();
}

public function obtenerLista ($userid='todos')
{
    $db = new eXist("admin", "admin");
    $db->connect();
    if (!strcmp($userid,'todos')){
        $query = "<songs>{let \$songs:=/dataSongs/songs/song for \$song in \$songs
        return <song><album>{\$song/album/text()}</album><track>{\$song/track/text()}</track></song>}</songs>";

    }
    else{
        $query = "<songs>{let \$songs:=/dataSongs/songs/song[userid = '".$userid."'] for \$song in \$songs
        return <song><album>{\$song/album/text()}</album><track>{\$song/track/text()}</track></song>}</songs>";
    }
    $db->setHighlight(FALSE);
    $result = $db->xquery($query);
    if ( !empty($result["XML"]) )
    {
        $res = $result["XML"];
        //Creating Instance of the Class
        $xmlObj    = new XmlToArray($res);
        //Creating Array
        $arrayData = $xmlObj->createArray();
        //print_r($arrayData['songs']);
    }
    $db->disconnect();
    return $arrayData['songs']['song'];
}

  public function obtenerCancion($album, $title){
      $db = new eXist("admin", "admin");
      $db->connect();
      $query = "let \$song:=/dataSongs/songs/song[album='".$album."' and track='".$title."']return <song>
      <artist>{\$song/artist/text()}</artist><album>{\$song/album/text()}</album><track>{\$song/track/text()}</track>
      <file>{\$song/file/text()}</file><genre>{\$song/genre/text()}</genre><year>{\$song/year/text()}</year></song>";
      $db->setHighlight(FALSE);
      $result = $db->xquery($query);
      if ( !empty($result["XML"]) )
      {
          $res = $result["XML"];
          $xmlObj    = new XmlToArray($res);
          $arrayData = $xmlObj->createArray();
      }
      $db->disconnect();
      //print_r($arrayData['song']);
      return $arrayData['song'];
  }
}
?>
```