Asignatura

Acceso a datos

UNIVERSAE
Instituto Superior de FP

Asignatura

Acceso a datos

UNIDAD 7

Base de datos de XML

UNIVERSAE
Instituto Superior de FP

XML con soporte de almacenamiento

Centrado en documentos

- Pocos elementos
- Gran cantidad de datos
- Sin estructura definida
- Procesamiento manual
- Ejemplo, Libros, Articulos, Informes, etc..

```
<?xml version="1.0" standalone="yes"?>
<articulos>
    <articulo categoria="tiempo">
        <titulo url=
        "https://www.investigacionyciencia.es/revistas/investigacion-y-ciencia/
        un-nuevo-plancton-737/colapso-rtico-16430">
                Colapso ártico
        </titulo>
        <fecha>11 de junio de 2018</fecha>
        <contenido>
                En 2003, veinticinco científicos tuvimos una revelación sobre el
                Ártico. La Fundación Nacional para la Ciencia de EE.UU. nos
                había invitado a celebrar un encuentro en Big Sky, Montana.
                Antes de esa reunión, cada uno de nosotros había limitado la
                investigación del Ártico a sus propios objetivos. Pero, al
                compartir nuestros estudios, llegamos a una inquietante
                conclusión: los cambios que habíamos observado por separado se
                relacionaban a la perfección entre sí. La totalidad del sistema
                ártico se dirigía hacia un nuevo estado de precariedad, y toda
                esperanza de detenerlo parecía improbable
        </contenido>
    </articulo>
</articulos>
```

Centrado en datos

- Muchos elementos
- Estructura definida
- Datos estructurados
- Procesamiento automatico
- Ejemplos, Facturas, Ficha de datos,

```
<?xml version="1.0">
<curso id="1">
    <nombre>Acceso a datos</nombre>
    <inicio>01/01/2022</inicio>
    <profesor>Antonio Roales Cabrasco</profesors>
    <aulas>
        <aula id="2">Aulario2</aula>
        <aula id="17">Taller informatica</aula>
    </aulas>
</curso>
```

TIPOS

Sistemas de ficheros

- Gestionar los ficheros XML mediante una jerarquía de carpetas.
- Las herramientas para buscar o gestionar son las que proporciona el sistema de archivos (Sistema operativo)

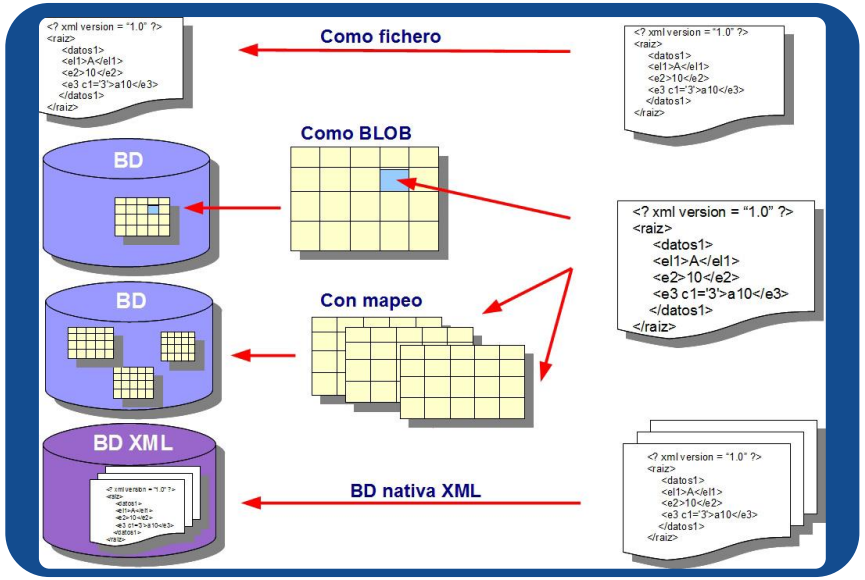
Bases de datos

- Con soporte a XML, puede ser del tipo relacional, orientada a objetos u objeto-relacional..
- Formas de almacenamiento
  - ➢ Una columna tipo binario (BLOB).
    - Se almacena el documento XML al completo.
    - El limitado, si el documento se tiene que modificar constantemente
  - ➢ Mapeo basado en tablas u objetos.
    - No se almacena el documento de forma intacta
    - Ajustar el documento XML a la estructura de la base de datos

Base de datos nativas de XML

- Optimizadas para XML.
- Se almacenan los documentos XML de forma intacta

Almacenamiento para documentos XML



Almacenamiento de XML en bases de datos relacionales

¿Por qué utilizar BBDD Relacionales para XML?

- Son las mas utilizadas
- SQL se ha actualizado para dar soporte a XML (SQL/XML)
    - Existe un tipo de datos XML
    - Uso de Xquery
    - Se puede generar documentos XML con el resultado de consultas.
- Aplican el soporte XML-compatible o XML-enabled
- Aplican el concepto de las transacciones sobre XML

Diferencias con las bases de datos nativas

Base de datos relacional	Base de datos nativas XML
Tiene su propio modelo de datos. Aplica una capa intermedia para dar soporte a xml	Modelo de datos propios
Solo pueden manejar y almacenar los documentos que encajan en su modelo	Manejan todos los tipos de documentos XML
Las consultas no se realizan sobre la estructura del documento	Las consultas se realizan sobre la misma estructura del documento

Características de las bases de datos de XML nativas

- **Colecciones**
  Formas de agrupar los documentos
- **Indexación**
  Mecanismos para la aceleración de búsquedas
- **Validación**
  Herramientas de validación, DTD o Esquemas
- **Transacciones**
  Integrar transacciones aún que su naturaleza dificulte su empleo
- **Consultas y modificaciones**
  Soporte de lenguajes como Xpath, Xquery y XSL
- **Soporte APIs externos**
  XQJ y XML:DB

¿Cuándo escoger una base de datos XML Nativa?

- ✓ Existen documentos XML con anidamientos profundos
- ✓ Preservar la integridad de los documentos
- ✓ Frecuencia de consultas de contenido

SGDB XML Nativas

Tamino

URL: https://documentation.softwareag.com/webmethods/tamino/tamino_vers.htm

MarkLogic

URL: https://www.marklogic.com/

existdb

URL: http://exist-db.org

BASE X

URL: https://basex.org/

sedna

URL: https://www.sedna.org/

eXist. Instalación

Instalación

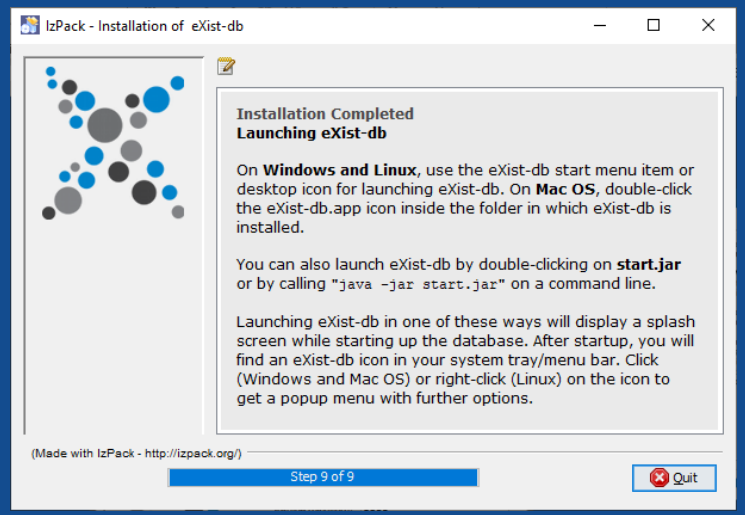
1. Descargar la distribución oficial 6.0.1 de su página web. *Exist-installer-6.0.1.jar*

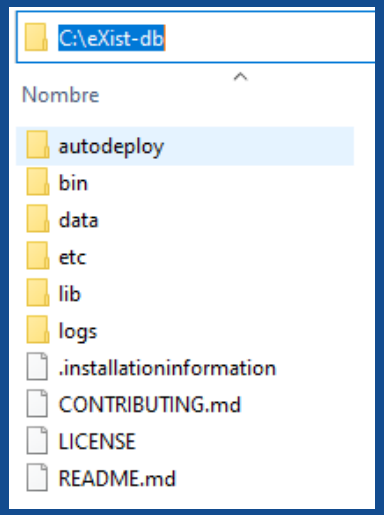
- Release Notes: https://exist-db.org/exist/apps/wiki/blogs/eXist/eXistdb601
- Maven Central: https://search.maven.org/search?q=g:org.exist-db

▾ Assets 6

eXist-db-6.0.1.dmg	148 MB
exist-distribution-6.0.1-unix.tar.bz2	147 MB
exist-distribution-6.0.1-win.zip	147 MB
exist-installer-6.0.1.jar	147 MB

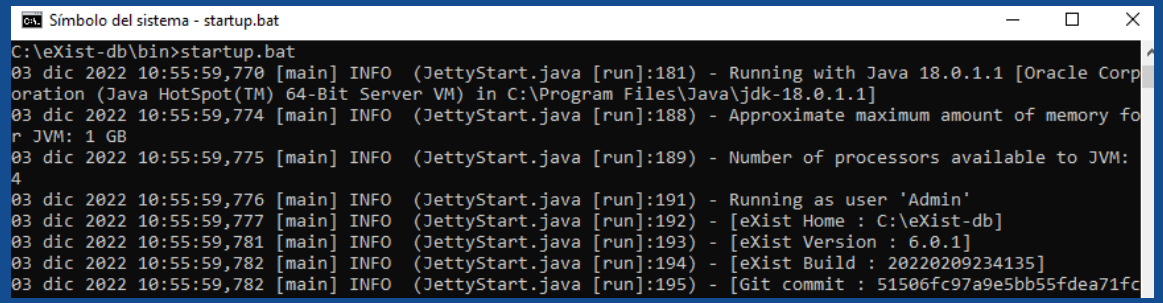
2. Ejecutar el asistente de instalación
3. Indicar la ruta donde se instalará la base de datos
4. Especificar una contraseña para el usuario admin

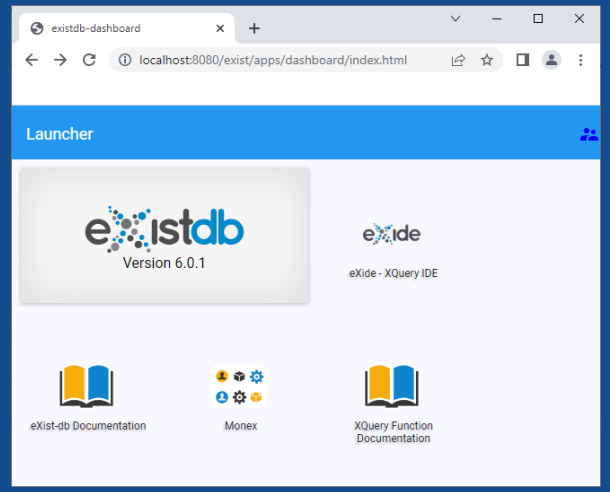




Arrancar el servidor

1. Abrir un terminal
2. Ubicarnos en la carpeta de instalación /bin
3. Ejecutar startup.bat
4. Acceder con el navegador. http://localhost:8080

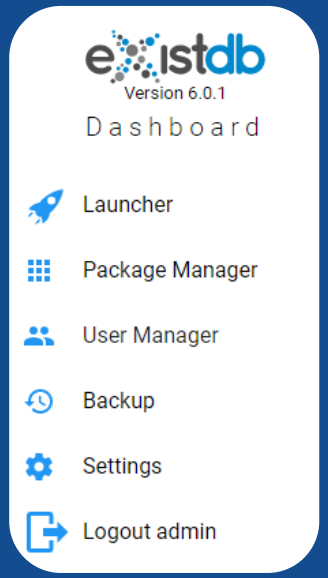


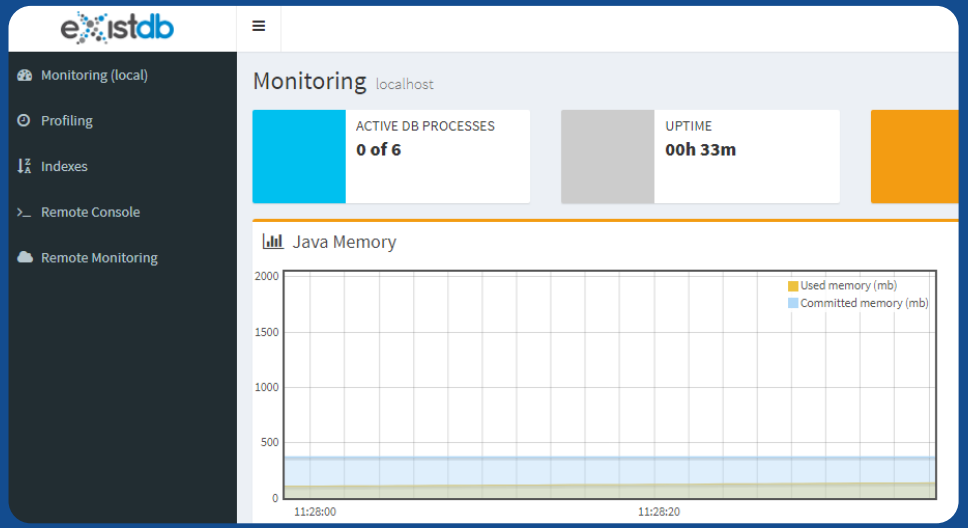


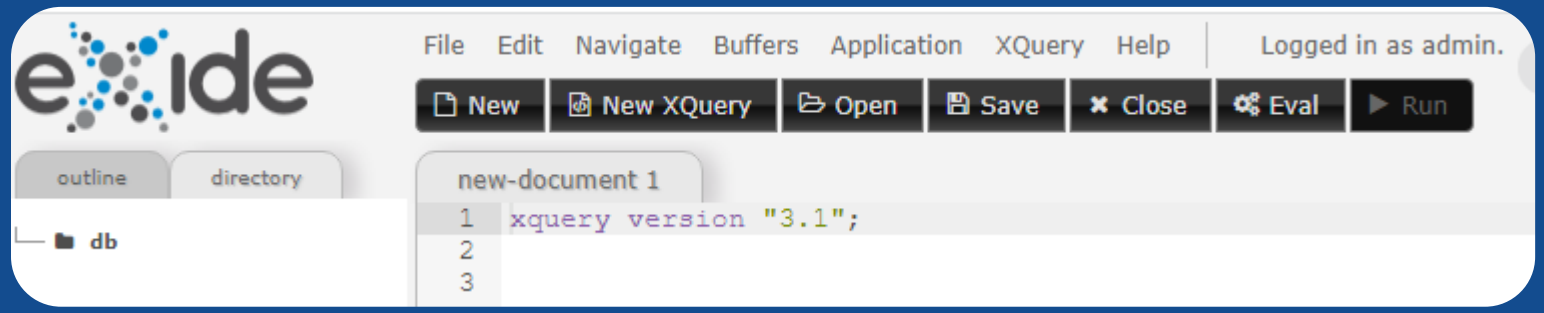
eXist. Panel de control

Opciones

- **Launcher**. Contiene las herramientas que se pueden utilizar en la base de datos
  - eXide. Un IDE para Xquery. Permite realizar consultas y navegar.
  - Monex. Permite monitorizar el servidor, consume de memoria, etc.
- **Package Manager**. Instalar paquetes y otras utilidades sobre la base de datos
- **User Manager.** Gestionar los usuarios de la base de datos
- **Backup**. Hacer copias de la base de datos.
- **Collections y Java Admin Client**. Estas opciones surgen en versiones anteriores o diferentes tipo de instalación

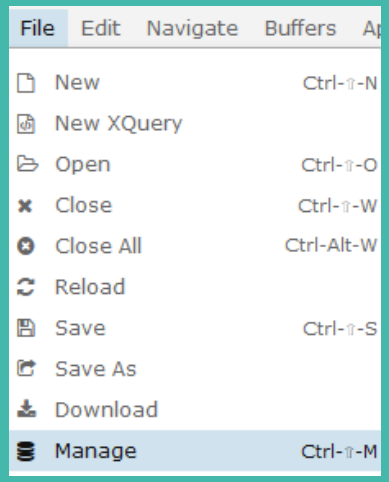


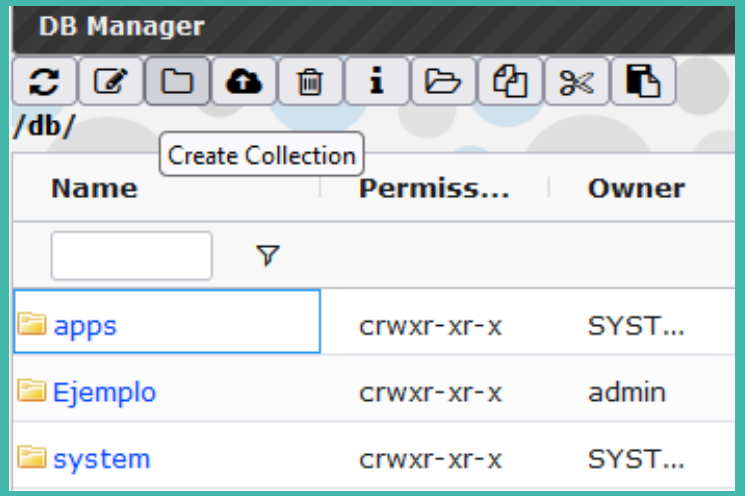




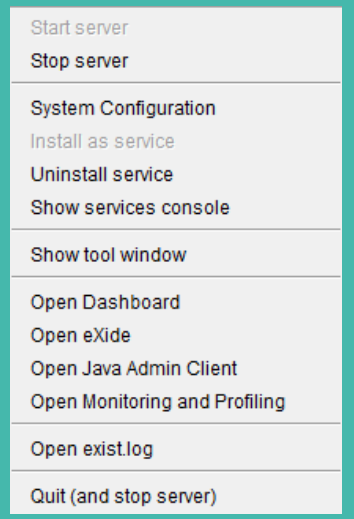
eXist. Administración

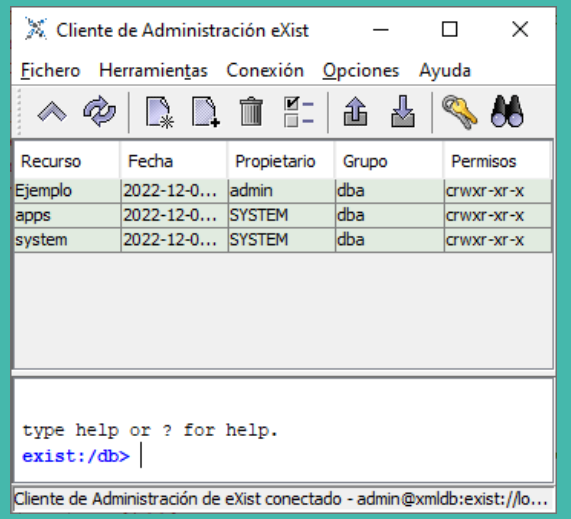
- DB Manager
  - Con el IDE eXide – File/Manage
  - Opciones de crear colecciones, subir documentos, copiar, pegar, ...





- Java Admin Client
  - Si la base de datos esta instalada como servicio.
  - Botón secundario sobre el icono de la barra de tareas de windows
  - Opciones de administrar usuarios, crear colecciones, realizar consultas, ...

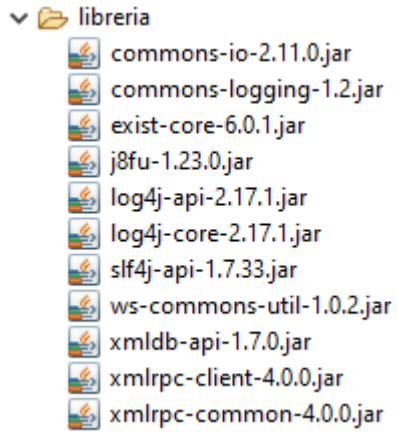




APIs para la gestión de la base de datos

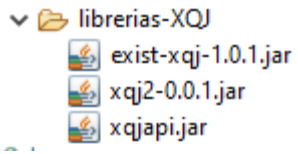
URL: *https://xmldb-org.sourceforge.net/xapi/*

- Interfaz para XML Nativas
- Permite realizar operaciones sobre colecciones
- Crear y borrar documentos XML
- Permite hacer uso de XQuery



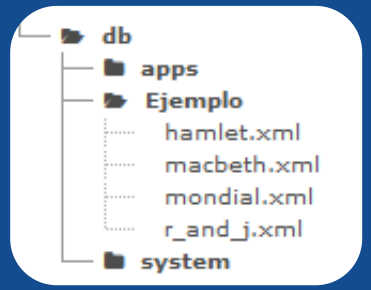
URL: *http://xqj.net/*

- XQuery for Java
- XQJ es la API para XML Nativas
- Parecido a JDBC para las bases de datos relacionales
- Permite realizar consultar y modificaciones sobre documentos XML



XML:DB. Acceso a la base de datos

```java
String driver = "org.exist.xmldb.DatabaseImpl";
String URI = "xmldb:exist://localhost:8080/exist/xmlrpc/db/";

// Inicializar la base de datos con el driver
Database database = (Database) Class.forName(driver).newInstance();
DatabaseManager.registerDatabase(database);

Collection collection = null;
XMLResource ficheroXML = null;
try {
    String coleccion = "Ejemplo";
    String fichero = "hamlet.xml";
    // Obtiene la colección
    collection = DatabaseManager.getCollection(URI + coleccion);

    if(collection != null) {
        // Obtiene el fichero
        ficheroXML = (XMLResource) collection.getResource(fichero);
        if (ficheroXML != null) {
            System.out.println(ficheroXML.getContent());
        } else {
            System.out.println("No se encuentra el fichero: "+fichero);
        }
    } else {
        System.out.println("No se encuentra la coleccion: "+coleccion);
    }
} catch (XMLDBException e) {
    e.printStackTrace();
} finally {

    if (ficheroXML != null)
        try { ((EXistResource) ficheroXML).freeResources(); }
        catch (XMLDBException xe) { xe.printStackTrace(); }

    if (collection != null)
        try { collection.close(); }
        catch (XMLDBException xe) { xe.printStackTrace(); }

}
```



new-document 1* | hamlet.xml

```
1  <?xml-stylesheet href="shakes.xsl" type="
2  <PLAY>
3      <TITLE>The Tragedy of Hamlet, Prince
4      <FM>
5          <P>ASCII text placed in the publi
6          <P>SGML markup by Jon Bosak, 199
7          <P>XML version by Jon Bosak, 199
8          <P>The XML markup in this versio
9  This work may freely be distributed on c
10 modified or altered in any way.</P>
11     </FM>
12     <PERSONAE>
13         <TITLE>Dramatis Personae</TITLE>
```

XML:DB. Crear y borrar colecciones

CREAR

```
String driver = "org.exist.xmldb.DatabaseImpl";
String URI = "xmldb:exist://localhost:8080/exist/xmlrpc/db/";
String usuario = "admin";
String password = "1234";

String collectionURI = "prueba";

Database database = (Database) Class.forName(driver).newInstance();
DatabaseManager.registerDatabase(database);

Collection coleccion = DatabaseManager.getCollection(URI + collectionURI);
if (coleccion == null) {
    Collection raiz = DatabaseManager.getCollection(URI, usuario, password);
    CollectionManagementService gestion =
            (CollectionManagementService) raiz.getService("CollectionManagementService", "1.0");
    coleccion = gestion.createCollection(collectionURI);
    coleccion.close();
    raiz.close();
} else {
    System.out.println("Ya existe la colección");
}
```

BORRAR

```
String driver = "org.exist.xmldb.DatabaseImpl";
String URI = "xmldb:exist://localhost:8080/exist/xmlrpc/db/";
String usuario = "admin";
String password = "1234";

String collectionURI = "prueba";

Database database = (Database) Class.forName(driver).newInstance();
DatabaseManager.registerDatabase(database);

Collection coleccion = DatabaseManager.getCollection(URI + collectionURI);
if (coleccion != null) {
    Collection raiz = DatabaseManager.getCollection(URI, usuario, password);
    CollectionManagementService gestion =
            (CollectionManagementService) raiz.getService("CollectionManagementService", "1.0");
    gestion.removeCollection(collectionURI);
    raiz.close();
} else {
    System.out.println("No existe la colección");
}
coleccion.close();
```

XML:DB. Crear y borrar documentos xml

CREAR

```
String driver = "org.exist.xmldb.DatabaseImpl";
String URI = "xmldb:exist://localhost:8080/exist/xmlrpc/db/";
String usuario = "admin";
String password = "1234";
String collectionURI = "prueba";

Database database = (Database) Class.forName(driver).newInstance();
DatabaseManager.registerDatabase(database);

// Obtiene la colección
Collection collection = DatabaseManager.getCollection(URI + collectionURI, usuario, password);

if (collection != null) {
    // Cargar un fichero existente
    String fichero = "src/empleados.xml";
    File file = new File(fichero);
    if (!file.canRead()) {
        System.out.println("No se puede leer el fichero " + fichero);
        return;
    }
    XMLResource resource = (XMLResource) collection.createResource(file.getName(), XMLResource.RESOURCE_TYPE);
    resource.setContent(file);
    collection.storeResource(resource);
    System.out.println("Documento guardado con id: " + resource.getId());

    // Generando un fichero
    resource = (XMLResource) collection.createResource("nuevo.xml", XMLResource.RESOURCE_TYPE);
    resource.setContent("<cliente>\n"
                        +"<nombre>Gonzalo</nombre>\n"
                        +"<apellido>Miró</apellido>\n"
                        +"</cliente>");
    collection.storeResource(resource);
    System.out.println("Documento guardado con id: " + resource.getId());

    ((EXistResource) resource).freeResources();

} else {
    System.out.println("No existe la colección");
}

collection.close();
```

BORRAR

```
String driver = "org.exist.xmldb.DatabaseImpl";
String URI = "xmldb:exist://localhost:8080/exist/xmlrpc/db/";
String usuario = "admin";
String password = "1234";
String collectionURI = "prueba";

Database database = (Database) Class.forName(driver).newInstance();
DatabaseManager.registerDatabase(database);

// Obtiene la colección
Collection collection = DatabaseManager.getCollection(URI + collectionURI, usuario, password);

if (collection != null) {

    XMLResource resource = (XMLResource) collection.getResource("Nuevo.xml");
    if(resource != null) {
        collection.removeResource(resource);
    } else {
        System.out.println("No existe el documento");
    }

    ((EXistResource) resource).freeResources();
} else {
    System.out.println("No existe la colección");
}

collection.close();
```

Lenguaje XQuery

XQuery

URL: *https://www.w3.org/TR/xquery/*

- Lenguaje de consultas sobre XML
- Basado en Xpath
- Lenguaje declarativo, parecido a SQL
- Xquery Update Facility es la extensión que permite la modificación de documentos

Estructuras de consulta

- Expresiones Xpath
- Expresiones FLWOR:
  - **For –** Selecciona nodos y los guarda en variables
  - **Let –** Asocia valores a variables (opcional)
  - **Where –** Filtrar los resultados (opcional)
  - **Order –** Ordenar los valores (opcional)
  - **Return –** Genera los valores de salida o devueltos

```
for $n in
doc('/db/ejemplos/Usuarios.xml')
/Usuarios/usuario
order by number ($n/CP)
return
<cli dni="{&n/string(@DNI)}"
nom="{&n/nombre}"></cli>
```

Estructuras de modificación

- **Insert** – Insertar uno o varios nodos
- **Delete** – Elimina nodos
- **Replace** – Modificar nodos
- **Rename**. – Renombra un nodo

```
Update insert
<usuario DNI="76534825F">
<fechareg>26-03-2022</fechareg>
</usuario>
Into
doc('/db/ejemplos/Usuarios.xml')/usuarios
```

Xquery. Ejemplos

new-document 1* | empleados.xml

```xml
<empleados>
    <empleado>
        <id>001</id>
        <nombre>Carolina</nombre>
        <apellido>Rodríguez</apellido>
        <email>carol001@email.com</email>
    </empleado>
    <empleado>
        <id>002</id>
        <nombre>Raúl</nombre>
        <apellido>Gonzalez</apellido>
        <email>ral002@email.com</email>
    </empleado>
    <empleado>
        <id>003</id>
        <nombre>Jesús</nombre>
        <apellido>Roldan</apellido>
        <email>jes003@email.com</email>
    </empleado>
</empleados>
```

```
xquery version "3.1";

for $elemento in doc('/db/prueba/empleados.xml')
return $elemento/empleados/empleado/nombre
```

__new__1

Adaptive Output | Indent | Live Preview

```
<nombre>Carolina</nombre>

<nombre>Raúl</nombre>

<nombre>Jesús</nombre>
```

```
xquery version "3.1";

for $elemento in doc('/db/prueba/empleados.xml')/empleados/empleado
where $elemento/id!="001"
order by $elemento/nombre
return <emp nombre="{$elemento/nombre}" email="{$elemento/email}"></emp>
```

__new__1

Adaptive Output | Indent | Live Preview | Highlight Index Matches

```
<emp nombre="Jesús" email="jes003@email.com"/>

<emp nombre="Raúl" email="ral002@email.com"/>
```

```
xquery version "3.1";

update insert
    <empleado>
        <id>004</id>
        <nombre>Esperanza</nombre>
        <apellido>Carrasco</apellido>
        <email>esp004@email.com</email>
    </empleado>
into doc('/db/prueba/empleados.xml')/empleados
```

```
xquery version "3.1";

for $emp in doc('/db/prueba/empleados.xml')/empleados/empleado
where $emp/email="esp004@email.com"
return update value $emp/email with "esperanza@email.com"
```

```
xquery version "3.1";

for $emp in doc('/db/prueba/empleados.xml')/empleados/empleado
where $emp/id='004'
return update delete $emp
```

new-document 1* | empleados.xml

```xml
<empleados>
    <empleado>
        <id>001</id>
        <nombre>Carolina</nombre>
        <apellido>Rodríguez</apellido>
        <email>carol001@email.com</email>
    </empleado>
    <empleado>
        <id>002</id>
        <nombre>Raúl</nombre>
        <apellido>Gonzalez</apellido>
        <email>ral002@email.com</email>
    </empleado>
    <empleado>
        <id>003</id>
        <nombre>Jesús</nombre>
        <apellido>Roldan</apellido>
        <email>jes003@email.com</email>
    </empleado>
<empleado>
        <id>004</id>
        <nombre>Esperanza</nombre>
        <apellido>Carrasco</apellido>
        <email>esperanza@email.com</email>
    </empleado>
</empleados>
```

XQJ. Acceso a la base de datos

```java
XQDataSource xqs = null;
XQConnection conexion = null;

try {
    xqs = new ExistXQDataSource();
    xqs.setProperty("serverName", "localhost");
    xqs.setProperty("port", "8080");
    xqs.setProperty("user","admin");
    xqs.setProperty("password","1234");

    conexion = xqs.getConnection();

    XQMetaData metadata = conexion.getMetaData();
    System.out.println("Conexión establecida como: "+metadata.getUserName());
    System.out.println("Versión XQJ: "+metadata.getXQJVersion());
    System.out.println("Driver:"+ metadata.getProductVersion());
    System.out.println("Transacciones: " + metadata.isTransactionSupported());

} catch(XQException e) {
    e.printStackTrace();
} finally {
    try {
        if (conexion != null)conexion.close();
    } catch (XQException e) {
        e.printStackTrace();
    }
}
```

Problems | Javadoc | Declaration | Console

<terminated> XQJ [Java Application] E:\Software\Eclipse\plugi

```
Conexión establecida como: admin
Transacciones: false
```

Librerias

- Descarga: http://xqj.net/exist/
- *exist-xqj-1.0.1.jar*, *xqj2-0.0.1.jar* y *xqjapi.jar*
- Paquetes:
  - javax.xml.xquery
  - net.xqj.exist.ExistXQDataSource

XQJ. Consultas y modificación

MODIFICACIÓN

```
XQDataSource xqs = null;
XQConnection conexion = null;

try {
    xqs = new ExistXQDataSource();
    xqs.setProperty("serverName", "localhost");
    xqs.setProperty("port", "8080");
    xqs.setProperty("user","admin");
    xqs.setProperty("password","1234");

    conexion = xqs.getConnection();

    String xquery = "update insert <empleado>"
                  +"<id>003</id>"
                  +"<nombre>Marta</nombre>"
                  +"<apellido>García</apellido>"
                  +"<email>mar003@email.com</email>"
                  +"</empleado>"
                  +" into doc('/db/prueba/empleados.xml')/empleados";

    String xquery2 = "update insert <empleado>"
                  +"<id>004</id>"
                  +"<nombre>Pedro</nombre>"
                  +"<apellido>Flores</apellido>"
                  +"<email>ped004@email.com</email>"
                  +"</empleado>"
                  +" following doc('/db/prueba/empleados.xml')/empleados/empleado/id[text()='003']/..";
    XQExpression xqe = conexion.createExpression();
    xqe.executeCommand(xquery);
    System.out.println("Se ha añadido un nuevo nodo: "+xquery);
    xqe.executeCommand(xquery2);
    System.out.println("Se ha añadido un nuevo nodo: "+xquery2);

} catch(XQException e) {
    e.printStackTrace();
} finally {
    try {
        if (conexion != null)conexion.close();
    } catch (XQException e) {
        e.printStackTrace();
    }
}
```

CONSULTAS

```
XQDataSource xqs = null;
XQConnection conexion = null;

try {
    xqs = new ExistXQDataSource();
    xqs.setProperty("serverName", "localhost");
    xqs.setProperty("port", "8080");
    xqs.setProperty("user","admin");
    xqs.setProperty("password","1234");

    conexion = xqs.getConnection();

    String xquery = "doc('/db/prueba/empleados.xml')/empleados/empleado";
    XQExpression xqe = conexion.createExpression();
    XQResultSequence rs = xqe.executeQuery(xquery);
    while (rs.next()) {
        System.out.println(rs.getItemAsString(null));
    }

} catch(XQException e) {
    e.printStackTrace();
} finally {
    try {
        if (conexion != null)conexion.close();
    } catch (XQException e) {
        e.printStackTrace();
    }
}
```

XQJ. Transacciones

Procedimiento.

- Establecer autocommit a falso
- Si todas las operaciones han ido bien, ejecutar .commit()
- Si se produce fallo, ejecutar .rollback()

```
XQDataSource xqs = null;
XQConnection conexion = null;

try {
    xqs = new ExistXQDataSource();
    xqs.setProperty("serverName", "localhost");
    xqs.setProperty("port", "8080");
    xqs.setProperty("user","");
    xqs.setProperty("password","");

    conexion = xqs.getConnection();
    conexion.setAutoCommit(false);

    String xquery = "";
    XQExpression xqe = conexion.createExpression();
    xqe.executeCommand(xquery);

    String xquery1 = "";
    xqe.executeCommand(xquery1);

    conexion.commit();
} catch(XQException e) {
    e.printStackTrace();
} finally {
    try {
        if (conexion != null) {
            conexion.rollback();
            conexion.close();
        }
    } catch (XQException e) {
        e.printStackTrace();
    }
}
```

Resumen

1. XML con soporte de almacenamiento
2. Almacenamiento para documentos XML
3. Almacenamiento de XML en base de datos relacionales
4. Características de las bases de datos de XML nativas
5. SGDB XML Nativas
6. eXist. Instalación
7. eXist. Panel de control
8. eXist. Administración
9. APIs para la gestión de la base de datos
10. XML:DB. Acceso a la base de datos
11. XML:DB. Crear y borrar colecciones
12. XML:DB. Crear y borrar documentos xml
13. El lenguaje Xquery
14. Xquery. Ejemplos
15. XQJ. Acceso a la base de datos
16. XQJ. Consultas y modificación
17. XQJ. Transacciones

UNIVERSAE

CHANGE YOUR WAY