


Open · Opened 3 days ago by  **Rubén Montero**

XQuery

Resumen

- Añadiremos un documento XML a nuestra base de datos *eXist*
- Escribiremos un programa en XQuery que consulta datos de dicho XML y produce una salida en XML

Descripción

El *Hello World* de nuestra tarea anterior no hace mucho:

```
let $message := 'Hello there'
return
<results>
  <message>{$message}</message>
</results>
```

1. Asigna una variable. Como ves, se usa `let` y `:=`
2. La emplea en una salida XML
3. Genera dicho XML. Se usa `return`

Entonces, XQuery sirve para **generar** documentos XML

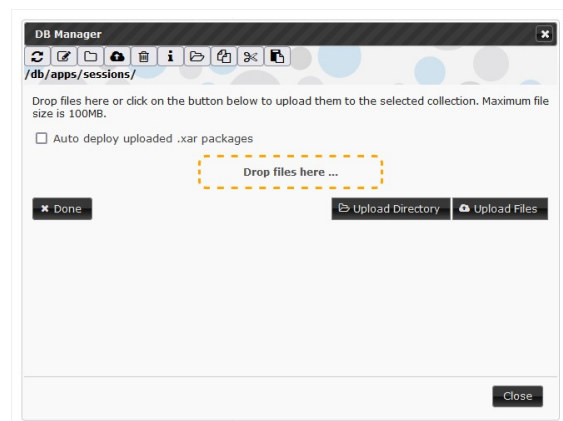
En realidad, se centra en **consultar** y **procesar** documentos XML

¿Sí?

Sí. Vamos a trabajar de esa manera.

Añade una nueva colección desde **eXide**. *File > Manage > Create collection*. Será `/db/apps/sessions/`.

Luego, pulsa el cuarto icono de la nube con una flecha que indica *Upload files*:



En nuestra nueva [colección sessions/](#) vamos a subir [este archivo XML](#). Descárgalo a tu ordenador (en realidad ya está en tu repositorio local) y **arrastralo** hasta el área *Drop files here*. Sucederá instantáneamente y no aparecerá una confirmación. Dale a *Close*.

Nuevo XQuery

Selecciona *New XQuery* para crear un programa nuevo.

Vamos a usar para ejemplificar la sintaxis de la sentencia **FLWOR** de Xquery

[FLOWR](#)

Es como el **SELECT, FROM, WHERE, ORDER BY** de SQL, pero se refiere a:

- **for** : Qué items se quieren seleccionar (*opcional*)
- **let** : Usado para crear variables temporales (*opcional*)

- **where** : Limitar los ítems seleccionados (*opcional*)
- **order** : Cambiar el orden de los resultados (*opcional*)
- **return** : Especifica la estructura de la salida (**obligatorio**)

En nuestro *Hello world!* hemos usado **let** y **return** .

Un ejemplo

Si trabajásemos con una *colección* **films** de un archivo XML como este:

```
<films>
  <film>
    <name>Pokemon: The Movie</name>
    <year>1998</year>
  </film>
  <film>
    <name>Lucario and the Mystery of Mew</name>
    <year>2005</year>
  </film>
</films>
```

Entonces, la siguiente **consulta XQuery**:

```
for $film in doc("films.xml")/films/film
  let $name := $film/name
  let $year := $film/year
  where $year = 1998
  return $name
```

...producirá esta **salida**:

```
<name>Pokemon: The Movie</name>
```

text()

En esta asignación:

```
let $name := $film/name
```

...se le está dando a **\$name** el valor del **nodo**. Es decir, **<name>...</name>** .

Si usamos **text()** , le daremos el valor del **contenido** (lo que haya *dentro* de **<name>...</name>**)

Dicho de otra forma, la anterior consulta es **equivalente a**:

```
for $film in doc("films.xml")/films/film
  let $name := $film/name/text()
  let $year := $film/year
  where $year = 1998
  return <name>{$name}</name>
```

Nuestro caso

Veamos un ejemplo dedicado a nuestro **sessions.xml** .

Este archivo contiene información de sesiones registradas en una página web donde los usuarios pueden visitar y leer libros.

Un ejemplo

En un *New XQuery*, pega la siguiente consulta:

```
for $session in doc("sessions.xml")/sessions/session
  let $username := $session/username/text()
  let $hour := $session/login/time/hour/text()
  let $minute := $session/login/time/minute/text()
  where $username = "Paul"
  return <time>{$hour}:{$minute}</time>
```

Tras esperar unos segundos para que **eXist** lo procese e indexe correctamente, podrás verificar que produce la siguiente salida:

```
<time>19:58</time>
<time>22:39</time>
<time>10:36</time>
<time>8:57</time>
<time>14:52</time>
<time>22:51</time>
<time>9:52</time>
<time>8:56</time>
```

order by

Para ordenar de forma ascendente por **hour**, modificaremos la consulta así:

```
for $session in doc("sessions.xml")/sessions/session
  let $username := $session/username/text()
  let $hour := $session/login/time/hour/number()
  let $minute := $session/login/time/minute/number()
  where $username = "Paul"
  order by $hour, $minute
  return <time>{$hour}:{$minute}</time>
```

Como ves:

- Se ha cambiado **text()** por **number()** en **\$hour** y **\$minute**. Así conseguimos que el contenido se interprete como número. Esto es necesario para que la ordenación sea correcta
- Se ha añadido **order by \$hour, \$minute**. Como ves, se puede especificar más de un criterio (campo) de ordenación. Así, si dos sesiones tiene la misma *hora*, se seguirá ordenando por *minuto*.

Elemento XML raíz

Si queremos que el **for** produzca resultados y se vean *dentro* de una etiqueta XML raíz, debemos *embeberlo* así:

```
<times>
{for $session in doc("sessions.xml")/sessions/session
  let $username := $session/username/text()
  let $hour := $session/login/time/hour/number()
  let $minute := $session/login/time/minute/number()
  where $username = "Paul"
  order by $hour, $minute
  return <time>{$hour}:{$minute}</time>
}
</times>
```

La tarea

Se pide que escribas una consulta **sessions.xql** que obtenga la información de *cuándo* fue accedido el libro **El inocente**:

- Devolverá un XML conformado por:
 - Elemento raíz **<bookAccess>**
 - Elemento **<name>El inocente</name>**
 - Elemento **<accessTimes>** que contenga:
 - Varios elementos **<access>**. Cada uno con:
 - **<viewedBy>**, contiene el *nombre de usuario*
 - **<date>**, contiene la fecha de acceso siguiendo el formato **dd/MM/yyyy**
- Sólo recuperará la información de los accesos que se produjeron al libro: **"El inocente"**
- Cada **<access>** estará ordenado de forma ascendente por el *nombre de usuario*.

Ejemplo para **EL principito**:

```
<bookAccess>
```

```
<name>El principito</name>
<accessTimes>
  <access>
    <viewedBy>Helena</viewedBy>
    <date>9/December/2010</date>
  </access>
  <access>
    <viewedBy>Helena</viewedBy>
    <date>20/August/2011</date>
  </access>
  ...
</accessTimes>
</bookAccess>
```

Una vez tengas tu consulta, **ejecútala** y genera el documento **en una nueva pestaña**.

Guarda `sessions.xql.xml` en `python-sessions/exist/results/` de tu repositorio.

Por último

Verifica que tu código pasa el *test* asociado a la tarea. Tendrás que ejecutar `test_017exist_flowr.py` usando Python.

Haz `commit` y `push` para subir los cambios al repositorio.



[Rubén Montero @ruben.montero](#) changed milestone to [%Sprint 5](#) 3 days ago