XML

- xQuery
- xPath

xQuery

XQuery

- Las instrucciones necesarias para iniciar el servidor eXists desde el terminal de Ubuntu.

```
cd /home/usuario/eXists/bin/
sudo ./startup.sh
En otro terminal y habiendo confirmado que el servidor se ha iniciado.
cd /home/usuario/eXists/bin/
sudo ./client.sh
```

- Crea una colección y dentro de ella otra dos.

```
mkcol Activitat2
cd Activitat2
exist:/db/Activitat2/> mkcol shakespeare
exist:/db/Activitat2/> mkcol varis
```

- Dentro de la colección shakespeare carga los archivos zip. Y dentro de la colección varios el archivo cotxes.xml

```
Cd shakespeare
Putzip /home/usuario/shakespeare.zip
cd ..
cd varis
put /home/usuario/*.xml
para ver el documento
edit cotxes.xml
```

- Haz una consulta para mostrar los diferentes modelos de coches, especificando el documento.

```
doc("cotxes.xml")/cotxes/cotxe/model/text()
```

- Utilizando el documento restaurant.xml muestra la descripción del segundo plato

```
for $x in doc("restaurant.xml")/restaurant/plat[2]/descripcio/text()
return $x/descripcio
```

- Haz una consulta sobre restaurant.xml sin especificar el documento y muestra el valor del atributo nombre del elemento opciones.

```
/restaurant/plat[@codi="12"]/opcions[@name]/text()
```

- Haz una consulta XQuery para mostrar los títulos y los artistas del los cds de USA, muestralo de la siguiente manera

```
<cds>
 <cd>
 <title>.....</title>
 <artist>.....</artist>
 </cd>
 <cd>
 <title>....</title>
 <artist>....</artist>
 </cd>
 ....
</cds>
```

Solución

```
<cds>
{
for $x in /catalog/cd[country="USA"]
let $titol := $x/title/text()
let $artist := $x/artist/text() order by $x/artist
return
<cd>
<artist>{$artist}</artist>
<titol>{$titol}</titol>
</cd>
}
</cds>
```

Fes una consulta en los documentos de shakespeare que se muestre de la siguiente manera

```
<obres>
 <obra>
  <titol>Títol de la obra de teatre</titol>
   <personatges>
    <nom>nom del personatge </nom>
    <nom>.....</nom>
        ......
   </personatges>
 </obra>
</obres>
```

Solución

```
<obres>{
for $x in /PLAY
```

```
let $titol := $x/TITLE/text()
return
<obra>
<title>{$titol}</title>
<personatges>
{
for $i in $x/PERSONAE//PERSONA
let $nom :=$i/text()
return
<nom>{$nom}</nom>
}
</personatges>
</obra>
}
</obres>
```

- Consulta XQuery en shakespeare que muestre todas los titulos de las obras y sus actos

```
<obres>
{
for $x in /PLAY
let $titol := $x/TITLE/text()
let $actes := count($x/ACT)
return
<obra>
<title>{$titol}</title>
<actes>{$actes}</actes>
</obra>
}
</obres>
```

xPath

XPath

Nota: Para ver los namespaces que corresponden a cada archivo. En la consola de Exist **put archivo.xml** o **put archivo.rdf**

- El elemento RDF que se encuentra en el primer nivel. Declaramos el namespace que contiene las reglas.

```
declare namespace rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#";
/rdf:RDF
```

- Selecciona todos los elementos Person

```
declare namespace foaf = "http://xmlns.com/foaf/0.1/";
//foaf:Person
```

- Selecciona todos los elemetos descendientes de foaf:Person

```
declare namespace foaf = "http://xmlns.com/foaf/0.1/";
//foaf:Person/*
```

- Selecciona de cada documento i de todos los elementos foaf:Person el primero que se encuentra encada documento.

```
declare namespace foaf = "http://xmlns.com/foaf/0.1/";
//foaf:Person[1]
```

- Selecciona de cada documento y el ultimo elemento foaf:Person

```
namespace foaf = "http://xmlns.com/foaf/0.1/";
//foaf:Person[last()]
```

- Selecciona todos los atributos rdf:nodeID de los elementos foaf:Person

```
declare namespace rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#";
declare namespace foaf = "http://xmlns.com/foaf/0.1/";
//foaf:Person/@rdf:nodeID
```

- Mostrar los elementos foaf:givenname que tienen atributo rdf:nodeID

```
declare namespace rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#";
declare namespace foaf = "http://xmlns.com/foaf/0.1/";
//foaf:Person[@rdf:nodeID]/foaf:givenname
```

- Seleccio elemeto por su id rdf:nodeID el valor 000003

```
declare namespace rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#";
declare namespace foaf = "http://xmlns.com/foaf/0.1/";
//foaf:Person[@rdf:nodeID=”000003”]
```

```
declare namespace rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#";
declare namespace foaf = "http://xmlns.com/foaf/0.1/";
//foaf:Person[@rdf:nodeID=”000003”]/foaf:givenname
```

- Mostrar foaf:givenname

```
declare namespace rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#";
declare namespace foaf = "http://xmlns.com/foaf/0.1/";
//foaf:Person[@rdf:nodeID=”000003”]/foaf:givenname/text()
```

- Mostrar elementos que tienen uno de estos intereres foaf:topic_interest “XML” o “BBDD”

```
declare namespace rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#";
declare namespace foaf = "http://xmlns.com/foaf/0.1/";
//foaf:Person[foaf:topic_interest=”XML" or foaf:topic_interest=”BBDD”]/foaf:givenname/text()
```

- Contar los elemento anteriores

```
declare namespace rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#";
declare namespace foaf = "http://xmlns.com/foaf/0.1/";
count(//foaf:Person[foaf:topic_interest=”XML” or foaf:topic_interest=”BBDD”])
```

- Devuelve una cadena de texto con todos los elementos de caso anterior

```
declare namespace rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#";
declare namespace foaf = "http://xmlns.com/foaf/0.1/";
```

```
string(//foaf:Person[foaf:topic_interest=”XML”  or foaf:topic_interest=”BBDD”])
```

- Devuelve un boolean

```
declare namespace rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#";
declare namespace foaf = "http://xmlns.com/foaf/0.1/";
boolean(//foaf:Person[foaf:topic_interest=”XML”  or foaf:topic_interest=”BBDD”])
```

- Devuelve los elementos que tiene menos de dos conocidos

```
declare namespace rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#";
declare namespace foaf = "http://xmlns.com/foaf/0.1/";
//foaf:Person/foaf:knows[count(foaf:Person)<2]
```

- Encuentra todos los elementos que la etiqueta comienza por “rdf”

```
declare namespace rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#";
declare namespace foaf = "http://xmlns.com/foaf/0.1/";
//*[startswith(name(),”rdf”)]
```

- Encuentra todos los elementos en los que la cadena del nombre contenga “knows”

```
declare namespace rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#";
declare namespace foaf = "http://xmlns.com/foaf/0.1/";
//*[contains(name(),”knows”)
```

- Buscar en tres documentos escogidos las personas interesadas en XML

```
declare namespace foaf = "http://xmlns.com/foaf/0.1/";
declare namespace rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#";
document('acortesg.rdf','jtallonp.rdf','jhiguerasa')//foaf:Person[foaf:topic_interest="XML"]/foaf:givenname/text()
```

- Retornar aquelles persones que el seu nom comença per J

```
declare namespace foaf = "http://xmlns.com/foaf/0.1/";
declare namespace rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#";
//foaf:Person/foaf:givenname[starts-with(text(),'J')]/text()
```