INDICE

EJEMPLOS

Ejemplo 1

```
doc("canciones.xml")/MiBibliotecaMP3/archivo/canción
```

1. Accede al documento XML "canciones.xml" y selecciona todos los elementos <canción> que están dentro de la estructura "MiBibliotecaMP3/archivo".

```
<canción>Hangar 18</canción>
<canción>Peace Sells</canción>
<canción>Master of Puppets</canción>
<canción>Among The Living</canción>
<canción>For Whom The Bell Tolls</canción>
```

Ejemplo 2

```
doc("canciones.xml")/MiBibliotecaMP3/archivo[puntuacion=10]
```

1. Accede al documento XML "canciones.xml" y selecciona el elemento <archivo> que contiene un subelemento <puntuación> igual a 10 dentro de la estructura <MiBibliotecaMP3>.

```
<archivo almacenado="DISCO1">
    <canción>Master of Puppets</canción>
    <artista>Metallica</artista>
    <disco>Master of Puppets</disco>
    <puntuacion>10</puntuacion>
    </archivo>
```

Ejemplo 3

```
1  doc("canciones.xml")/MiBibliotecaMP3/archivo[puntuacion>8]/canción
```

1. Accede al documento XML "canciones.xml" y selecciona todos los elementos <canción> que están dentro de los elementos <archivo> que tienen una puntuación mayor a 8 en la estructura <MiBibliotecaMP3>

```
1  <canción>Hangar 18</canción>
2  <canción>Peace Sells</canción>
3  <canción>Master of Puppets</canción>
4
```

Ejemplo 4

```
for $i in doc("canciones.xml")/MiBibliotecaMP3/archivo
where $i/puntuacion>8
return $i/canción
```

1.Asigna cada elemento <archivo> seleccionado a la variable $i. Esto permite iterar sobre todos los elementos <archivo> en la estructura especificada.

2."where" filtra los elementos <archivo> en función de la <puntuación> verificando si es mayor a 8.

3. Devuelve el contenido de los elementos <canción> que están dentro de los elementos <archivo> que cumplen la condición anterior.

```
<canción>Hangar 18</canción>
<canción>Peace Sells</canción>
<canción>Master of Puppets</canción>
```

Ejemplo 5

```
<html>
<head>
<title>Ejemplo 5</title>
</head>

<body>
<ol>
{
for $i in doc("canciones.xml")/MiBibliotecaMP3/archivo
where $i/puntuacion>8
order by $i/puntuacion
return <li> {$i/canción}({$i/puntuacion}) </li>
}
</ol>
</body>
</html>
```

7. Inicio de una lista ordenada.

9. Asigna cada elemento <archivo> seleccionado a la variable $i. Esto permite iterar sobre todos los elementos <archivo> en la estructura especificada.

10. "where" filtra los elementos <archivo> en función de la <puntuación> verificando si es mayor 8

11.”order by” ordena los elementos <archivo> en función de la puntuación de forma ascendente.

12. Devuelve un elemento <li> para cada subelemento <cancion> y <puntuacion> del elemento <archivo> que cumple con las condiciones anteriores.

14. Fin de una lista ordenada

```
<html><head><title>Ejemplo 5</title></head><body><ol><li><canción>Master of Puppets</canción>(<puntuacion>10</puntuacion>) </li>
```

Ejemplo 6

```
1  <html>
2  <head>
3  <title>Ejemplo 6</title>
4  </head>
5
6  <body>
7  <ol>
8  {
9  for $i in doc("canciones.xml")/MiBibliotecaMP3/archivo
10 where $i/puntuacion>8
11 order by $i/puntuacion
12 return <li> {data($i/canción)}({data($i/puntuacion)}) </li>
13 }
14 </ol>
15 </body>
16 </html>
```

7. Inicio de una lista ordenada.

9. Asigna cada elemento <archivo> seleccionado a la variable $i. Esto permite iterar sobre todos los elementos <archivo> en la estructura especificada.

10. "where" filtra los elementos <archivo> en función de la <puntuación> verificando si es mayor 8

11. ”order by” ordena los elementos <archivo> en función de la puntuación de forma ascendente.

12. Devuelve un elemento <li> para cada contenido de <cancion> y <puntuacion> del elemento <archivo> que cumple con las condiciones anteriores.

14. Fin de una lista ordenada

1. Master of Puppets(10)
2. Hangar 18(9)
3. Peace Sells(9)

Ejemplo 7

```
<html>
<head>
<title>Ejemplo 7</title>
</head>
<body>
<ol>
{
for $i in doc("canciones.xml")/MiBibliotecaMP3/archivo
where $i/puntuacion>8
order by $i/puntuacion
return <li class="{data($i/@almacenado)}"> {data($i/canción)}({data($i/puntuacion)}) </li>
}
</ol>
</body>
</html>
```

6. Inicio de una lista ordenada.

8. Asigna cada elemento <archivo> seleccionado a la variable $i. Esto permite iterar sobre todos los elementos <archivo> en la estructura especificada.

9. "where" filtra los elementos <archivo> en función de la <puntuación> verificando si es mayor 8

10. ”order by” ordena los elementos <archivo> en función de la puntuación de forma ascendente.

11. Devuelve un elemento <li> para contenido de los subelementos <cancion> y <puntuacion> de cada elemento <archivo> que cumple con las condiciones anteriores. La clase del elemento <li> se establece utilizando el contenido del atributo “almacenado” del elemento <archivo> actual.

13. Fin de una lista ordenada

```
<html><head><title>Ejemplo 7</title></head><body><ol><li class="DISCO1">Master of Puppets(10) </li>
```

Ejemplo 8 (Tabla)

```
<html>
<head>
<title>Ejemplo 8</title>
</head>

<body>
<table border = "1">
<caption>DISCO 1 </caption>
<tr><td>Artista</td><td>Disco</td></tr>
{
for $i in doc("canciones.xml")/MiBibliotecaMP3/archivo
order by $i/puntuacion
return if ($i/@almacenado="DISCO1")
then <tr><td>{data($i/artista)}</td><td>{data($i/disco)}</td></tr>
else ()
}
</table>
<br/>
<table border = "1">
<caption>DISCO 2 </caption>
<tr><td>Artista</td><td>Disco</td></tr>
{
for $i in doc("canciones.xml")/MiBibliotecaMP3/archivo
order by $i/puntuacion
return if ($i/@almacenado="DISCO2")
then <tr><td>{data($i/artista)}</td><td>{data($i/disco)}</td></tr>
else ()
}
</table>
</body>
</html>
```

7. Inicio de la primera tabla. Se establece el atributo "border" para mostrar los bordes de la tabla.

8. Establece el título de la primera tabla como "DISCO 1"

9. Define la primera fila de la tabla con los encabezados "Artista" y "Disco".

11. Asigna cada elemento <archivo> seleccionado a la variable $i. Esto permite iterar sobre todos los elementos <archivo> en la estructura especificada.

12. ”order by” ordena los elementos <archivo> en función de la puntuación de forma ascendente.

13. Devuelve si el atributo "almacenado" del elemento <archivo> actual tiene el valor "DISCO1"

14. Si es verdadero, genera una fila en la tabla con el contenido de <artista> y <disco>

15. Si es falso, se devuelve un resultado vacío ().

17. Fin de la primera tabla.

DISCO 1

Artista	Disco
Metallica	Master of Puppets
Metallica	Ride The Lightning
Megadeth	Rust in Peace

Ejemplo 9 (Tabla)

```
<html>
<head>
<title>Ejemplo 9</title>
</head>

<body>
<table border="1">
<caption>CANCIONES POR DISCO </caption>
<tr><td>Artista</td><td>Nombre disco</td><td>Grabada en</td></tr>
{
for $i in doc("canciones.xml")/MiBibliotecaMP3/archivo
order by $i/puntuacion
return if ($i/@almacenado="DISCO1")
then <tr><td>{data($i/artista)}</td><td>{data($i/disco)}</td>
     <td>DISCO1</td></tr>
else <tr><td>{data($i/artista)}</td><td>{data($i/disco)}</td>
     <td>DISCO2</td></tr>
}
</table>
</body>
</html>
```

7. Inicio de la tabla. Se establece el atributo "border" para mostrar los bordes de la tabla.

8. Establece el título de la primera tabla como "CANCIONES POR DISCO"

9. Define la primera fila de la tabla con los encabezados "Artista", "Nombre disco" y "Grabada en".

11. Asigna cada elemento <archivo> seleccionado a la variable $i. Esto permite iterar sobre todos los elementos <archivo> en la estructura especificada.

12. ”order by” ordena los elementos <archivo> en función de <puntuación> de forma ascendente.

13. Devuelve si el atributo "almacenado" del elemento <archivo> actual tiene el valor "DISCO1"

14. Si es verdadero, se genera una fila de la tabla con el nombre del artista, el nombre del disco y la etiqueta "DISCO1" en la columna "Grabada en".

16. Si es falso, se genera una fila de la tabla con el nombre del artista, el nombre del disco y la etiqueta "DISCO2" en la columna "Grabada en".

CANCIONES POR DISCO

Artista	Nombre disco	Grabada en
Metallica	Master of Puppets	DISCO1
Anthrax	Among The Living	DISCO2
Metallica	Ride The Lightning	DISCO1
Megadeth	Rust in Peace	DISCO1
Megadeth	Peace Sells...But Who's Buying	DISCO2

Ejemplo 10 (Tabla)

```
<html>
<head>
<title>Ejemplo 10</title>
</head>

<body>
<table>
<caption>CANCIONES DE METALLICA </caption>
<tr><td>Canción</td><td>Disco</td><td>Grabada en</td></tr>
{
for $i in doc("canciones.xml")/MiBibliotecaMP3/archivo
where $i/artista="Metallica"
order by $i/puntuacion
return <tr><td>{data($i/artista)}</td><td>{data($i/disco)}</td>
     <td>{data($i/@almacenado)}</td></tr>
}
</table>
</body>
</html>
```

7. Inicio de la tabla. Se establece el atributo "border" para mostrar los bordes de la tabla.

8. Establece el título de la tabla como "CANCIONES DE METALLICA".

9. Define la primera fila de la tabla con los encabezados "Canción", "Disco" y "Grabada en".

11. Asigna cada elemento <archivo> seleccionado a la variable $i. Esto permite iterar sobre todos los elementos <archivo> en la estructura especificada.

12. “where” filtra los elementos <archivo> para aquellos cuyo elemento <artista> sea igual a "Metallica"

13. ”order by” ordena los elementos <archivo> en función de <puntuación> de forma ascendente.

14. Devuelve una fila de la tabla con el contenido de <artista>, <disco> y “almacenado” obtenidos de los elementos correspondientes del elemento <archivo> actual.

17. Fin de la tabla.

CANCIONES DE METALLICA

Canción	Disco	Grabada en
Metallica	Master of Puppets	DISCO1
Metallica	Ride The Lightning	DISCO1

Ejemplo 12

```
1  <html>
2  <head>
3  <title>Ejemplo 12</title>
4  </head>
5
6  <body>
7  <ul>
8  {
9  for $i at $j in doc("canciones.xml")/MiBibliotecaMP3/archivo
10 where $i/puntuacion>8
11 order by $i/puntuacion
12 return <li>{$j}. {data($i/canción)}({data($i/puntuacion)}) </li>
13 }
14 </ul>
15 </body>
16 </html>
```

7. Inicio de la lista desordenada.

9. "for" para iterar sobre cada elemento <archivo> dentro de la estructura "MiBibliotecaMP3/archivo" en el documento XML "canciones.xml". La variable $i representa cada elemento <archivo> durante la iteración, y la variable $j representa el índice de la iteración actual.

10. "where" filtra los elementos <archivo> en función de la <puntuación> verificando si es mayor 8

11. ”order by” ordena los elementos <archivo> en función de <puntuación> de forma ascendente.

12. Devuelve un elemento de lista <li> para cada contenido de <cancion> y <puntuacion> del elemento <archivo> que cumple las condiciones anteriores. El contenido del elemento de lista incluye el índice ($j), el nombre de la canción (obtenido del elemento <canción>) y la puntuación (obtenida del elemento <puntuacion>).

14. Fin de la lista desorndeada

- 3. Master of Puppets(10)
- 1. Hangar 18(9)
- 2. Peace Sells(9)

Ejemplo 18 (Tabla)

```
1  <html>
2  <head>
3  <title>Ejemplo 18</title>
4  </head>
5
6  <body>
7  <table border="1">
8  <caption>CANCIONES POR DISCO </caption>
9  <tr><td>Artista</td><td>Nombre</td><td>Grabada en</td></tr>
10 {
11 for $i in doc("canciones.xml")/MiBibliotecaMP3/archivo
12 order by $i/artista ascending, $i/canción descending
13 return <tr><td>{data($i/artista)}</td><td>{data($i/canción)}</td>
14      <td>{data($i/@almacenado)}</td></tr>
15 }
16 </table>
17 </body>
18 </html>
```

7. Inicio de la tabla. Se establece el atributo "border" para mostrar los bordes de la tabla.

8. Establece el título de la tabla como "CANCIONES POR DISCO".

9. Define la primera fila de la tabla con los encabezados "Canción", "Disco" y "Grabada en".

11. Asigna cada elemento <archivo> seleccionado a la variable $i. Esto permite iterar sobre todos los elementos <archivo> en la estructura especificada.

12. Ordena los elementos <archivo> en función del elemento <artista> de forma ascendente y cuando se repite el <artista> se ordenará en función del elemento <canción> de forma descendente.

13. Devuelve una fila de la tabla con el contenido de <artista>, <disco> y “almacenado” obtenidos de los elementos correspondientes del elemento <archivo> actual.

16. Fin de la tabla.

CANCIONES POR DISCO

Artista	Nombre	Grabada en
Anthrax	Among The Living	DISCO2
Megadeth	Peace Sells	DISCO2
Megadeth	Hangar 18	DISCO1
Metallica	Master of Puppets	DISCO1
Metallica	For Whom The Bell Tolls	DISCO1

Ejercicio 19 (Tabla)

```
<html>
<head>
<title>Ejemplo 19</title>
</head>

<body>
<table border="1">
<caption>CANCIONES POR DISCO </caption>
<tr><td>Artista</td><td>Nombre</td><td>Grabada en</td></tr>
{
for $i in doc("canciones.xml")/MiBibliotecaMP3/archivo
let $numero := (substring($i/@almacenado,6,1))
order by $i/artista ascending, $i/canción descending
return <tr><td>{upper-case(data($i/artista))}</td><td>{data($i/canción)}</td>
     <td>{$numero}</td></tr>
}
</table>
</body>
</html>
```

7.Inicio de la tabla. Se establece el atributo "border" para mostrar los bordes de la tabla.

8. Establece el título de la tabla como "CANCIONES POR DISCO".

9. Define la primera fila de la tabla con los encabezados "Canción", "Disco" y "Grabada en".

11. Asigna cada elemento <archivo> seleccionado a la variable $i. Esto permite iterar sobre todos los elementos <archivo> en la estructura especificada.

12. Se declara una variable $numero que obtiene un subconjunto de caracteres del atributo "almacenado" del elemento <archivo>. El subconjunto comienza en el sexto carácter y tiene una longitud de un carácter.

13. Ordena los elementos <archivo> en función del elemento <artista> de forma ascendente y cuando se repite el <artista> se ordenará en función del elemento <canción> de forma descendente.

14. Devuelve una fila de la tabla con los valores del artista, canción y número obtenidos de los elementos correspondientes del elemento "archivo" actual. El nombre del artista se convierte a mayúsculas usando la función "upper-case()".

17.Fin de la tabla.

CANCIONES POR DISCO

Artista	Nombre	Grabada en
ANTHRAX	Among The Living	2
MEGADETH	Peace Sells	2
MEGADETH	Hangar 18	1
METALLICA	Master of Puppets	1
METALLICA	For Whom The Bell Tolls	1

EJERCICIOS

Ejercicio 1

```
for $x in doc("libros.xml")/biblioteca/libros/libro
return <libro>{$x/titulo, $x/editorial}</libro>
```

1.Asigna cada elemento <libro> seleccionado a la variable $x. Esto permite iterar sobre todos los elementos <libro> en la estructura especificada.

2. Devuelve un nuevo elemento <libro> que contiene los subelementos <titulo> y <editorial> de cada <libro> encontrado.

```
<libro>
  <titulo>Learning XML</titulo>
  <editorial>O'Reilly</editorial>
</libro>
<libro>
  <titulo>XML Imprescindible</titulo>
  <editorial>O'Reilly</editorial>
</libro>
<libro>
  <titulo>XML Schema</titulo>
  <editorial>O'Reilly</editorial>
</libro>
<libro>
  <titulo>XPath Essentials</titulo>
  <editorial>Wiley</editorial>
</libro>
```

Ejercicio 2

```
for $x in doc("libros.xml")/biblioteca/libros/libro
where number($x/paginas) < 100
return data($x/titulo)
```

1.Asigna cada elemento <libro> seleccionado a la variable $x. Esto permite iterar sobre todos los elementos <libro> en la estructura especificada.

2. "where" filtra los elementos <libro> en función del número de páginas. La expresión number($x/paginas) convierte el contenido del elemento <paginas> en un número, y lo compara para verificar si es menor a 100.

3. Devuelve el contenido de los <titulo> de los <libro> que cumplan la condición anterior.

```
Learning XML
```

Ejercicio 3

```
1 for $x in doc("libros.xml")/biblioteca/libros
2 let $y := $x/libro[number(paginas) < 100]
3 return count($y)
```

1. Busca los elementos <libros> en el documento XML "libros.xml",

2. Filtra los libros que tienen menos de 100 páginas y la cláusula **"let"** asigna a la variable **$y** los elementos <libro> que cumplen la condición.

3. Devuelve la cantidad de libros en la variable **$y**.

```
1
```

Ejercicio 4

```
1  <ul>
2  {
3  for $x in doc("libros.xml")/biblioteca/libros/libro
4  where $x/editorial = "O'Reilly"
5  order by $x/titulo
6  return <li>{data($x/titulo)}</li>
7  }
8  </ul>
```

1. Inicia la creación de una lista HTML ordenada.

3. "for" que asigna cada elemento <libro> seleccionado a la variable $x. Esto permite iterar sobre todos los elementos <libro> en la estructura especificada.

4. "where" filtra los elementos <libro> en función de su editorial. En este caso <editorial> si es igual a "O'Reilly".

5. Ordena los libros en función del contenido del elemento "titulo" de manera ascendente.

6. Devuelve un elemento de lista HTML (<li>) creado con los contenido de <título> de cada <libro>.

8.Cierra la lista HTML ordenada.

```
1  <ul>
2    <li>Learning XML</li>
3    <li>XML Imprescindible</li>
4    <li>XML Schema</li>
5    <li>XQuery</li>
6  </ul>
```

Ejercicio 5

```
for $x in doc("libros.xml")/biblioteca/libros/libro
where $x[number(@publicacion)=2002]
return <libro>{$x/titulo, $x/editorial}</libro>
```

1.Asigna cada elemento <libro> seleccionado a la variable $x. Esto permite iterar sobre todos los elementos <libro> en la estructura especificada.

2."where" filtra los elementos "libro" en función del valor del atributo "publicacion" y verifica si es un valor numérico igual a 2002.

3. Devuelve un nuevo elemento <libro> que contiene los subelementos <titulo> y <editorial> de los <libro> que cumplan la condición anterior.

```
<libro><titulo>XML Schema</titulo><editorial>O'Reilly</editorial></libro>
<libro><titulo>XPath Essentials</titulo><editorial>Wiley</editorial></libro>
```

Ejercicio 6

```
for $x in doc("libros.xml")/biblioteca/libros/libro
where count($x/autor)>1
return <libro>{$x/titulo, $x/editorial}</libro>
```

1.Asigna cada elemento <libro> seleccionado a la variable $x. Esto permite iterar sobre todos los elementos <libro> en la estructura especificada.

2."where" filtra los elementos <libro> con la expresión count($x/autor) la cual cuenta cuántos elementos <autor> están presentes dentro del elemento <libro> actual, y la condición mayor a 1.

3. Devuelve un nuevo elemento <libro> que contiene los subelementos <titulo> y <editorial> de cada <libro> que cumplieron la condición anterior.

```
<libro>
  <titulo>XML Imprescindible</titulo>
  <editorial>O'Reilly</editorial>
</libro>
```

Ejercicio 7

```
for $x in doc("libros.xml")/biblioteca/libros/libro
where $x/versionElectronica
return <libro>{$x/titulo, $x/editorial}</libro>
```

1. Asigna cada elemento <libro> seleccionado a la variable $x. Esto permite iterar sobre todos los elementos <libro> en la estructura especificada.

2. "where" filtra los elementos <libro> en función de la presencia del elemento <versionElectronica> dentro de cada <libro>. Si está presente, cumple la condición.

3. Devuelve un nuevo elemento <libro> que contiene los subelementos <titulo> y <editorial> de cada libro que cumple la condición anterior.

```
<libro>
  <titulo>Learning XML</titulo>
  <editorial>O'Reilly</editorial>
</libro>
<libro>
  <titulo>XPath Essentials</titulo>
  <editorial>Wiley</editorial>
</libro>
```

Ejercicio 8

```
for $x in doc("libros.xml")/biblioteca/libros/libro
where empty ($x/versionElectronica)
return $x/titulo
```

1.Asigna cada elemento <libro> seleccionado a la variable $x. Esto permite iterar sobre todos los elementos <libro> en la estructura especificada.

2. "where" filtra los elementos <libro> en función de la ausencia del elemento <versionElectronica> dentro de cada <libro>. Si el elemento no existe, el libro cumple la condición.

3. Devuelve el subelemento <titulo> del elemento <libro> que cumplan la condición anterior.

```
<titulo>XML Imprescindible</titulo>
<titulo>XML Schema</titulo>
<titulo> Beginning XSLT 2.0: Form Novice to Professional</titulo>
<titulo> XQuery</titulo>
```