**INDICE**

[EJEMPLOS 2](#__RefHeading___Toc153_3558397977)

[Ejemplo 1 2](#__RefHeading___Toc155_3558397977)

[Ejemplo 2 3](#__RefHeading___Toc157_3558397977)

[Ejemplo 3 4](#__RefHeading___Toc159_3558397977)

[Ejemplo 4 5](#__RefHeading___Toc161_3558397977)

[Ejemplo 5 6](#__RefHeading___Toc163_3558397977)

[Ejemplo 6 7](#__RefHeading___Toc165_3558397977)

[Ejemplo 7 8](#__RefHeading___Toc167_3558397977)

[Ejemplo 8 (Tabla) 9](#__RefHeading___Toc169_3558397977)

[Ejemplo 9 (Tabla) 10](#__RefHeading___Toc171_3558397977)

[Ejemplo 10 (Tabla) 11](#__RefHeading___Toc173_3558397977)

[Ejemplo 12 12](#__RefHeading___Toc516_3558397977)

[Ejemplo 18 (Tabla) 13](#__RefHeading___Toc175_3558397977)

[Ejercicio 19 (Tabla) 14](#__RefHeading___Toc177_3558397977)

[EJERCICIOS 15](#__RefHeading___Toc179_3558397977)

[Ejercicio 1 15](#__RefHeading___Toc393_3558397977)

[Ejercicio 2 16](#__RefHeading___Toc395_3558397977)

[Ejercicio 3 17](#__RefHeading___Toc397_3558397977)

[Ejercicio 4 18](#__RefHeading___Toc399_3558397977)

[Ejercicio 5 19](#__RefHeading___Toc401_3558397977)

[Ejercicio 6 20](#__RefHeading___Toc403_3558397977)

[Ejercicio 7 21](#__RefHeading___Toc405_3558397977)

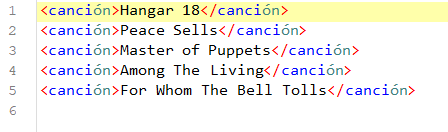
[Ejercicio 8 22](#__RefHeading___Toc407_3558397977)

# **EJEMPLOS**

## Ejemplo 1



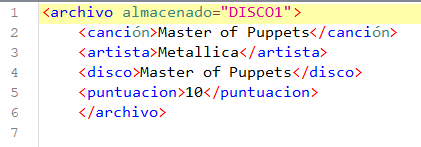
1. Accede al documento XML "canciones.xml" y selecciona todos los elementos <canción> que están dentro de la estructura "MiBibliotecaMP3/archivo".



## Ejemplo 2



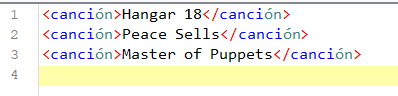
1. Accede al documento XML "canciones.xml" y selecciona el elemento <archivo> que contiene un subelemento <puntuación> igual a 10 dentro de la estructura <MiBibliotecaMP3>.



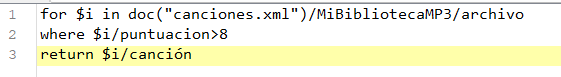
## Ejemplo 3



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>



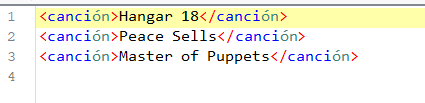
## Ejemplo 4



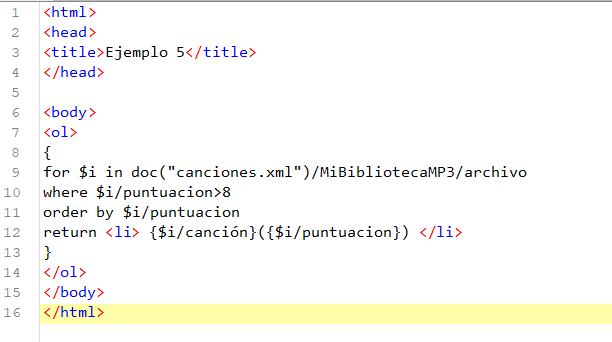
1.Asigna cada elemento <archivo> seleccionado a la variable $i. Esto permite iterar sobre todos los elementos <archivo> en la estructura especificada.

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.



## Ejemplo 5



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

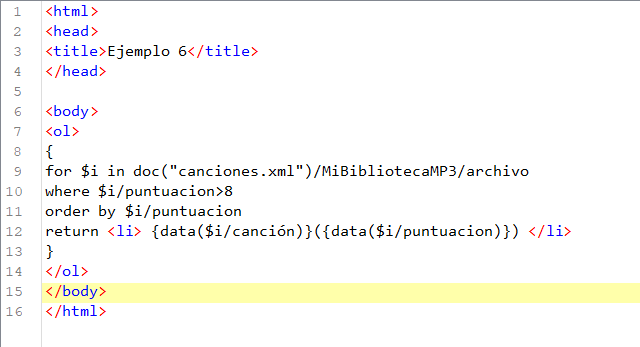
11.”order by” ordena los elementos <archivo> en función de la puntuación de forma ascendente.

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



# Ejemplo 6



7. Inicio de una lista ordenada.

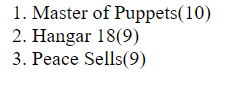
9. Asigna cada elemento <archivo> seleccionado a la variable $i. Esto permite iterar sobre todos los elementos <archivo> en la estructura especificada.

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



## Ejemplo 7



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

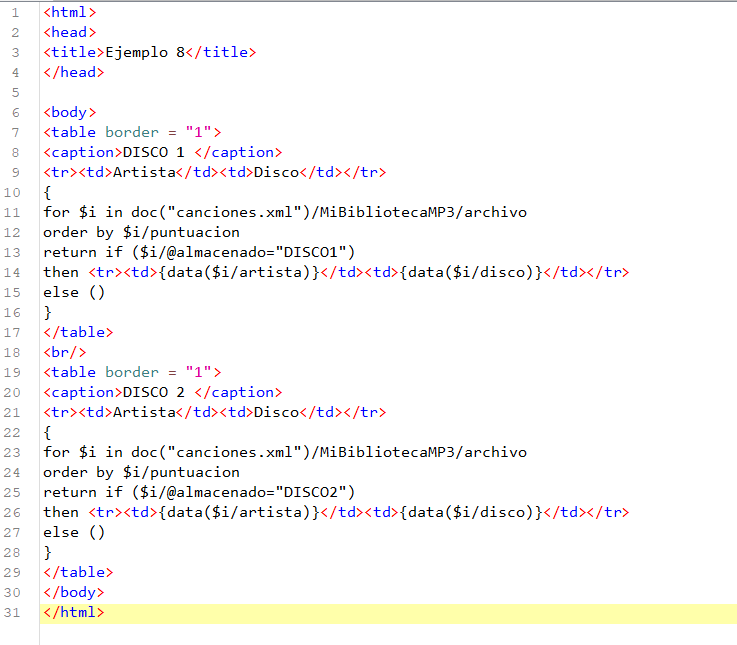
10. ”order by” ordena los elementos <archivo> en función de la puntuación de forma ascendente.

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



## Ejemplo 8 (Tabla)



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.

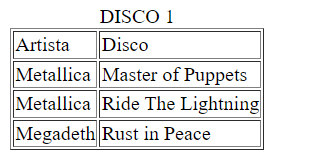
12. ”order by” ordena los elementos <archivo> en función de la puntuación de forma ascendente.

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

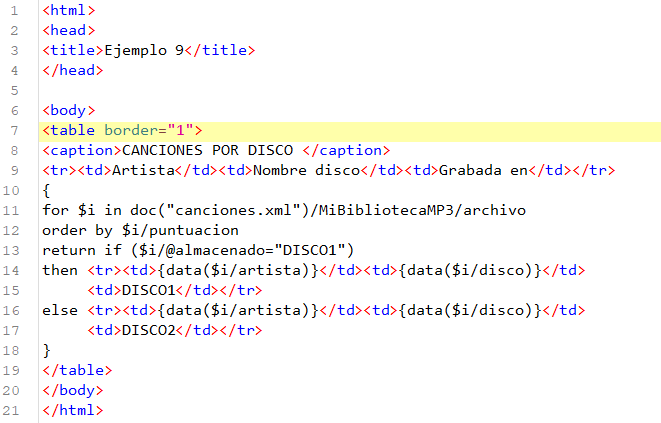
14. Si es verdadero, genera una fila en la tabla con el contenido de <artista> y <disco>

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

17. Fin de la primera tabla.



## Ejemplo 9 (Tabla)



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".

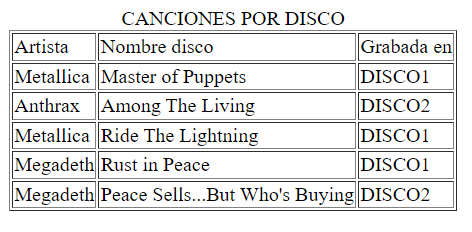
11. Asigna cada elemento <archivo> seleccionado a la variable $i. Esto permite iterar sobre todos los elementos <archivo> en la estructura especificada.

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

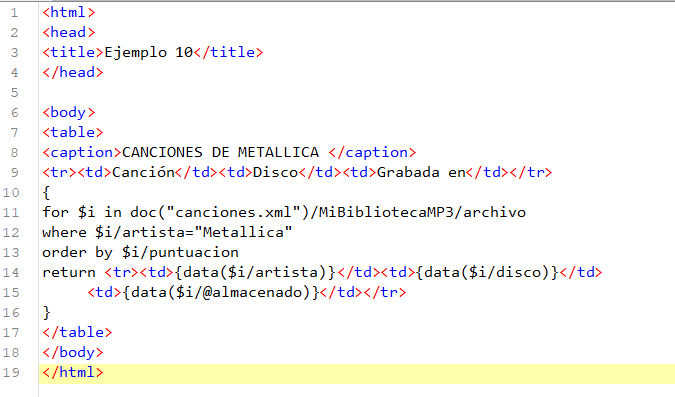
13. Devuelve si el atributo "almacenado" del elemento <archivo> actual tiene el valor "DISCO1"

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".



## Ejemplo 10 (Tabla)



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".

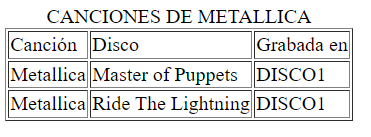
11. Asigna cada elemento <archivo> seleccionado a la variable $i. Esto permite iterar sobre todos los elementos <archivo> en la estructura especificada.

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

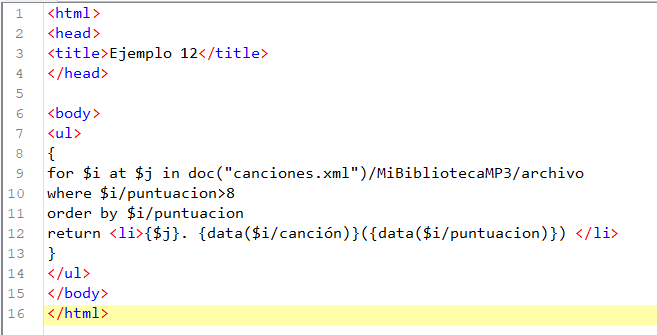
13. ”order by” ordena los elementos <archivo> en función de <puntuación> de forma ascendente.

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.



## Ejemplo 12



7. Inicio de la lista desordenada.

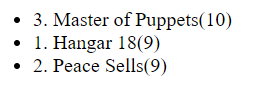
9. "for" para iterar sobre cada elemento <archivo> dentro de la estructura "MiBibliotecaMP3/archivo" en el documento XML "canciones.xml". La variable $i representa cada elemento <archivo> durante la iteración, y la variable $j representa el índice de la iteración actual.

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

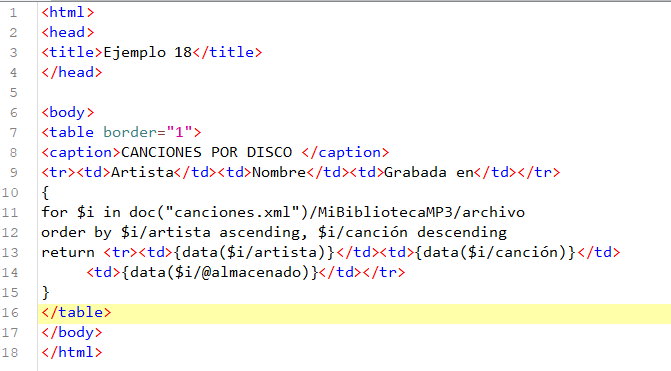
11. ”order by” ordena los elementos <archivo> en función de <puntuación> de forma ascendente.

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



## Eje**mplo** 18 (Tabla)



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.



## Ejercicio 19 (Tabla)



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.

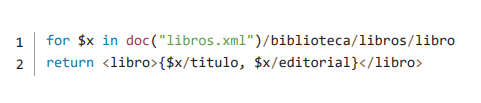
14. Devuelve una fila de la tabla con los valores del artista, canción y número obtenidos de los elementos correspondientes del elemento "archivo" actual. El nombre del artista se convierte a mayúsculas usando la función "upper-case()".

17.Fin de la tabla.



# EJERCICIOS

## Ejercicio 1



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.



## Ejercicio 2



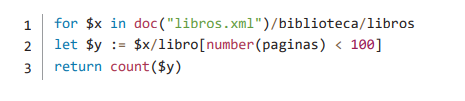
1.Asigna cada elemento <libro> seleccionado a la variable $x. Esto permite iterar sobre todos los elementos <libro> en la estructura especificada.

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.



## Ejercicio 3



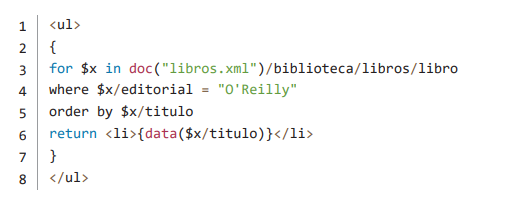
1. Busca los elementos <libros> en el documento XML "libros.xml",

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**.



## Ejercicio 4



1. Inicia la creación de una lista HTML ordenada.

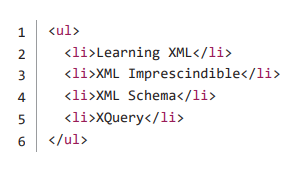
3. "for" que asigna cada elemento <libro> seleccionado a la variable $x. Esto permite iterar sobre todos los elementos <libro> en la estructura especificada.

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.



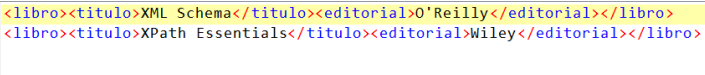
## Ejercicio 5

## 

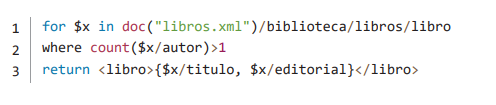
1.Asigna cada elemento <libro> seleccionado a la variable $x. Esto permite iterar sobre todos los elementos <libro> en la estructura especificada.

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.



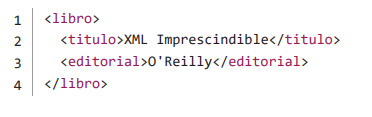
## Ejercicio 6



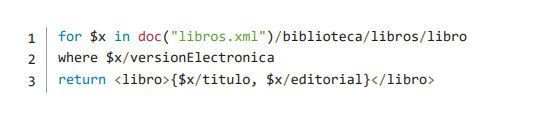
1.Asigna cada elemento <libro> seleccionado a la variable $x. Esto permite iterar sobre todos los elementos <libro> en la estructura especificada.

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.



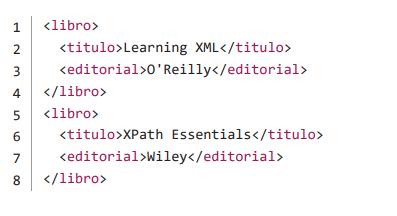
## Ejercicio 7



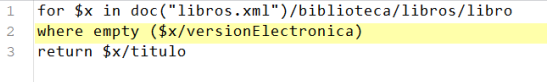
1. Asigna cada elemento <libro> seleccionado a la variable $x. Esto permite iterar sobre todos los elementos <libro> en la estructura especificada.

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.



## Ejercicio 8



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.

