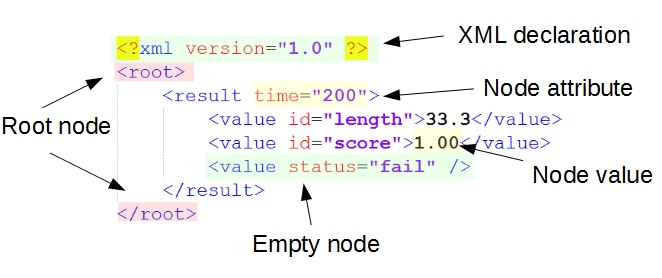
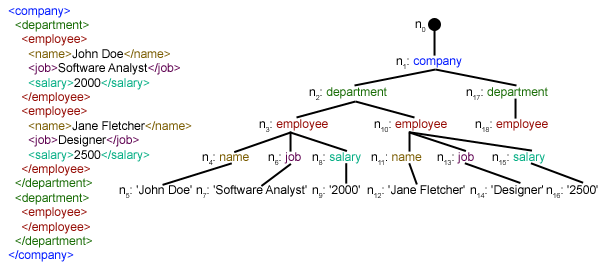
## ACCESO LOCAL: XML (I)

El formato **XML** (eXtensible Markup Language) es parecido al HTML, pero es más estructurado y es un tipo de formato de fichero que tiene cierto uso, y a diferencia del json, es más probable que encuentres datos que te interesen en este formato. [Ojo que eso no le quita importancia al Json].

### Estructura de un archivo XML



Además de lo visto para el ejemplo de arriba sobre etiquetas, una forma de ver los archivos XML es como si formasen un estructura de arbol en el que van apareciendo conjuntos de valores que pertenencen a otros conjuntos de valores a partir de un nodo raíz:



### Lectura de ficheros XML

Antes de utilizar la correspondiente librería de python (ojo esta no se llama xml) que nos ayude a procesar un fichero XML y extraer la información que pueda contener, veamos uno como texto plano que es:

[1]:

**with** open("./data/cd\_catalog.xml","r")**as** f:

**for** line **in** f:

print(line,end**=**"")

Texto

Descripción generada automáticamente

Y si hacemos un pequeño ejercicio de interpretar las etiquetas llegaríamos a un árbol como este:

Diagrama, Forma, Polígono

Descripción generada automáticamente

Lo que podría interesarnos es sacar los datos por CD para construirnos una tabla Catalog y ahí poner una fila por cd con su identificador (que podría ser nuestro índice de DataFrame por ejemplo) y los valores de cada una de esas etiquetas (TITLE, ARTIST, etc) las columnas... ¿Y cómo hacerlo?

Lo vamos a hacer con ayuda de la librería ElementTree:

[2]:

**import** xml.etree.ElementTree **as** ET

Lo primero es leer y parsear el fichero de esta forma:

[4]:

tree**=**ET.parse("./data/cd\_catalog.xml")

### Procesando un fichero XML

Esta librería trata el XML como si fuese un árbol. En este formato de árbol, disponemos de diversos métodos con los que podemos extraer partes del XML.

* tag muestra el texto dentro de la etiqueta
* attrib muestra los atributos de la etiqueta
* text muestra el texto del nodo
* La función iter() permite conocer la estructura del XML
* La función find() busca en el XML y devuelve el elemento que coincide con la etiqueta especificada.
* La función findall() devuelve todos los elementos con cierta etiqueta

Una vez cargado juguemos con los métodos anteriores para obtener la información o los datos útiles de nuestro catálogo de CDs:

[5]:

*# Obtener la etiqueta del nodo raiz:*

raiz**=**tree.getroot()

raiz.tag

[5]:

'CATALOG'

Bien, poco hemos avanzado, porque esto ya lo intuíamos, vamos más allá y capturemos para cada elemento el nombre de sus etiquetas (aquí sólo tenemos CD como elemento, en el caso de la figura anterior teníamos DEPARTMENT, EMPLOYEE)

[6]:

*# Para cada elemento sus etiquetas*

**for** elemento **in** raiz.iter():

print(elemento.tag)

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Esto lo aplana un poco y nos dice mucho sobre la jerarquía pero podemos ver jugar un poco para ver la estructura de forma más jerarquica además de ver los atributos y los valores (si hubiera unos u otros):

[10]:

*# Recorrido con cierta jerarquia*

**for** hijo **in** raiz:

tabs**=**"\t"

print(hijo.tag,hijo.attrib,hijo.text)

**for** nieto **in** hijo:

print(tabs,nieto.tag,nieto.text)

Además puedo buscar elementos por sus etiquetas y hacer lo anterior de otra forma

[13]:

cds**=**tree.findall("CD")

**for** cd **in** cds:

print("Id:",cd.attrib["id"])

print("Titulo:",cd.find("TITLE").text)

print("Artista:",cd.find("ARTIST").text)

Texto

Descripción generada automáticamente

Usando cualquiera de las dos formas anteriores podría recorrer el XML y crear una estructura con la que luego obtener un DataFrame, pero eso lo veremos en la siguiente sesión.