OutputFormat

Esta clase especifica un formato de salida, básicamente establece una serie de normas para indentar, codificar… Se asigna, por ejemplo a un objeto XMLSerializer. A continuación se detallan algunas de las formas de uso más utilizadas.

# Establecer tamaño de línea

Mediante el método **setLineWidth(int size)** podremos establecer el tamaño de la línea antes de que se haga un corte con la indentación, esto sirve para controlar las líneas largas del fichero antes de que hagan un “wrap” o rompan en una nueva.

# Establecer el separador de línea

Por defecto será (\n), pero si queremos cambiarlo podremos utilizar el método **setLineSeparator(String separator)** para cambiarlo por ejemplo a (\n\r) y que sea más compatible.

# Establecer la indentación o tabulación del fichero

Para activarlo usaremos el método **setIndenting(boolean true)** y luego especificaremos el número de espacios con el método **setIndent(int indent)** que queremos utilizar. La indentación nos sirve para mantener nuestro fichero tabulado y ordenado en base a nodos padres e hijos, no es algo determinante a la hora de hacerlo funcionar, en XML, en otros lenguajes como Python es obligatorio indentar, de lo contrario el código no funcionará. Hay que tener en cuenta que el fichero ocupa más espacio cuando está indentado, así que si estamos buscando esta premisa lo mejor es minificarlo o eliminar todo tipo de espacios posibles.

# Establecer la codificación de caracteres

Mediante el método **setEncoding(String encoding)** especificaremos el conjunto de caracteres que nuestro fichero reconocerá en base a los estándares, como por ejemplo **setEncoding("UTF-8")** y **setEncoding("ISO-8859-1").**

# Establecer elementos fuera de codificación

Mediante **setNonEscapingElements(String[] elements)** especificaremos los elementos que pueden contener estos 5 elementos:

" &quot;

' &apos;

< &lt;

> &gt;

& &amp;

Digamos que son caracteres reservados que deben convertirse a su formato de la derecha para que nuestro documento siga bien formado y sea válido.

# Establecer el tipode multimedia del fichero

Una vez más, en base a unos estándares, podremos establecer el tipo de datos que el fichero contiene, el MediaType es un formato nacido para reconocer los tipos de fichero en internet, con el método **setMediaType(String type)** podrémos especificarlo. Para XML el formato será application/xml aunque también puede ser text/xml y por ejemplo, para HTML sería text/html. Todo esto está definido por la W3C.

# Establecer el tipo de documento

Mediante el método **setDocType(String publicId, String systemId)** especificaremos los ámbitos públicos y de sistema de nuestro fichero, en base a unos estándares de la W3C. Sólo será necesario si el DOM Document y los eventos SAX no están especificados en los tipos de media.

# Establecer la versión del formato de salida

Mediante el método **setVersion(String version)** especificaremos la versión del formato de salida, para XML debería ser 1.0 y para HTML 4.0

# Establecer la declaración de XML

Para que un documento sea identificado como XML debería empezar con una declaración como <?xml version="1.0" encoding="UTF-8"?>. Mediante el método **setOmitXMLDeclaration(boolean false)** le diremos si queremos que esto sea así o no.

# Establecer si el fichero es autónomo o depende de fuentes de datos externas

En la declaración de un XML (la de arriba) se puede especificar si un documento es autónomo por sí mismo o si se basa en fuentes externas de datos. Una declaración de un fichero que no es autónomo por sí mismo sería algo como <?xml version="1.0" encoding="UTF-8" standalone="yes"?>. Mediante el método **setStandalone(boolean true)** podremos especificar este parámetro en la declaración del XML.

# Establecer elementos representados con CDATA

El CDATA en XML sirve para expresar un tipos de dato de texto que no es representado como texto XML o no debería, nos sirve para incluir textos con formateo u otros tipo de datos, lo mejor es ver un ejemplo de lo que es un dato representado con CDATA. Esto nos ayudará a representar datos extraños y que nuestro documento siga bien formado (well formed).

<exampleOfACDATA>

<![CDATA[

Since this is a CDATA section

I can use all sorts of reserved characters

like > < " and &

or write things like

<foo></bar>

but my document is still well formed!

]]>

</exampleOfACDATA>

Mediante el método setCDataElements(String[] elements) podremos especificar cuáles elementos (por nombre) se representarán de esta manera en el formato de salida.

# Omitir comentarios

Podremos ignorar los comentarios del documento XML mediante el método **setOmitComments(boolean true)**.