**Informe de Laboratorio PTI – XML Processing with Java**

Sergio Mazzariol

## Acerca de XML

XML, (eXtensible Markup Language), es un meta-lenguaje que permite definir lenguajes de marcas desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible. Proviene del lenguaje SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML) para estructurar documentos grandes. A diferencia de otros lenguajes, XML da soporte a bases de datos, siendo útil cuando varias aplicaciones deben comunicarse entre sí o integrar información.

Busca dar solución al problema de expresar información estructurada de la manera más abstracta y reutilizable posible. Que la información sea estructurada quiere decir que se compone de partes bien definidas, y que esas partes se componen a su vez de otras partes. Entonces se tiene un árbol de trozos de información. Como por ejemplo una librería, que se compone de libros, que a su vez tienen autores, esta parte se llaman elementos y se señalan mediante etiquetas.

Hoy en día XML es un lenguaje ampliamente utilizado, hoy realizaremos una aplicación de alquiler de coches, donde podremos guardar los datos de cada solicitud de alquiler en un fichero XML, del cual escribiremos y leeremos para poder listar todos los coches rentados hasta el momento.

Para esta practica se partió de los archivos de ejemplo de gitlab, partiendo de allí se crearon los comandos de:

**Reset**, crea un nuevo documento XML en memoria, luego la guara en un fichero carrental.xml. Si el archivo ya existe, su contenido anterior se perderá

**New,** añade un nuevo elemento “rental” al documento, este elemento posee, marca, modelo, número de días, número de unidades, descuento, esto se añade al documento carrental.xml

**List,** lee el fichero carrental.xml en memoria y lo imprime por pantalla.

**Xslt,** lee el fichero carrental.xml en memoria, lo transforma a HTML con una hoja de estilo XSLT y lo imprime por consola.

**Validate,** lee el fichero carrental.xml en memoria, lo valida con carrental.xsd.

Para preparar el entorno de trabajo de la practica, he actualizado la versión de java y realizado un export para generar una variable que es util a la hora de compilar.

export CLASSPATH=./xalan.jar:./xercesImpl.jar:./jdom.jar:.

Ahora podemos construir nuestros ficheros Java con el comando

Javac Fichero.java

Seguidamente para ejecutarlo, hacemos

Java Example

## Otras opciones posibles

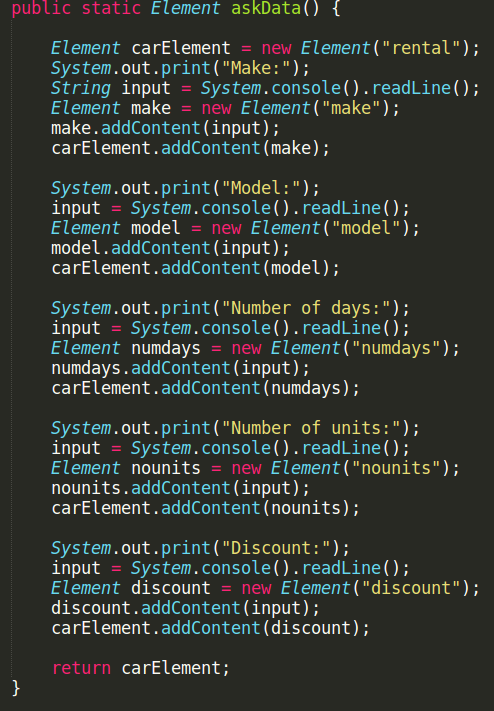
XML es un lenguaje que hoy en día esta muy extendido, aunque JSON este ganando terreno rápidamente, ya que por ejemplo en JSON no se puede hacer una validación de datos como en XML. Hay otras alternativas existentes quizás no tan conocidas, como lo son YAML, AXON, OGDL, ConfigObj, entre muchos otros. Pero hoy en día el que se utiliza más ampliamente es XML.

Para crear, leer, modificar archivos XML no solo es posible con Java, también hay otros lenguajes de programación que pueden realizar estas acciones, como lo son PHP, Python, JavaScript, entre otros.

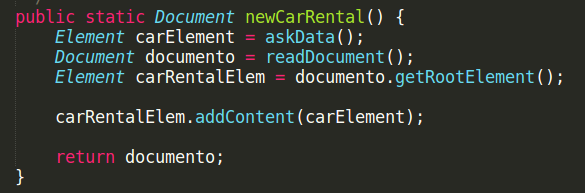
## Solución encontrada

Al crear las funcionalidades solicitadas, con un de las que tuve más inconvenientes fue con la de **new,** en esta funcionalidad se solicitaba añadir un nuevo rental al documento XML, para esto se tenían que leer los datos a través de la consola de comandos.

Primero creamos el elemento base de “rental”, seguido a esto creamos el primer elemento de “rental” que en este caso es “make”, luego lo leemos por consola, con este valor obtenido lo añadimos al elemento “make”, por ultimo añadimos el elemento “make” al elemento “rental” y así para cada atributo del rental. En la siguiente captura se puede ver con más detalle el procedimiento.

****

Luego de que ya hemos creado el elemento rental, procedemos a añadirlo al elemento raíz, para esto hemos leído el documento, al cual le pedimos obtener el roo Element con la función getRootElement, por últimos añadimos el rental creado anterior mente a nuestro elemento raíz.

****

Otra función difícil de programar, fue la de “validate”, para esta función no había suficiente información en el tutorial y poca información en Internet, primero se debía añadir el schema cuando se crea el documento nuevo, esto se hace con Namespaces, se crea un nuevo Namespace el cual luego se añade como atributo al elemento raíz del documento.



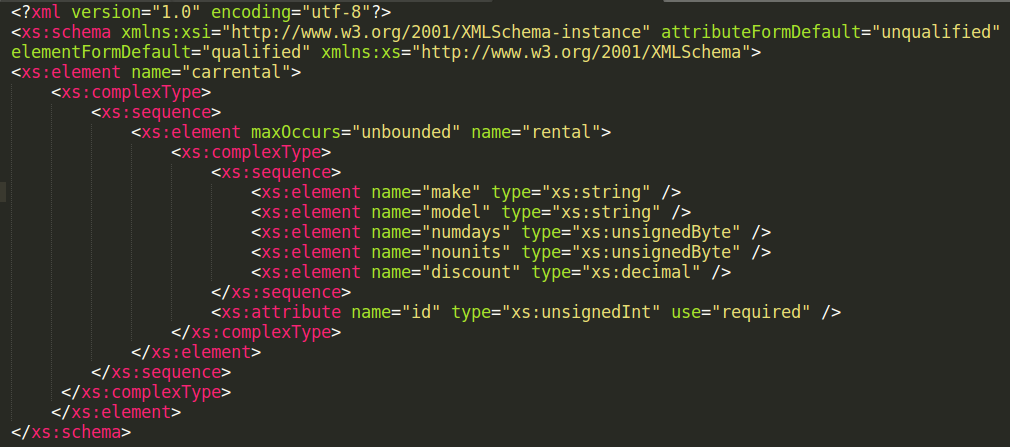
Luego de tener correctos los atributos del elemento raiz, procedemos a realizar la validación, esta se hace llamando a otro fichero el cual se llama carrental.xsd, este fichero contiene todas las normas que nuestro documento XML debe serguir, si no cumple alguna de estas, un error se mostrará por pantalla.



En este documento xsd, podemos definir el tipo de variable para cada elemento, estas son algunas de las más utilizadas:

|  |  |
| --- | --- |
| Name | Description |
| byte | A signed 8-bit integer |
| decimal | A decimal value |
| int | A signed 32-bit integer |
| integer | An integer value |
| long | A signed 64-bit integer |
| negativeInteger | An integer containing only negative values (..,-2,-1) |
| nonNegativeInteger | An integer containing only non-negative values (0,1,2,..) |
| nonPositiveInteger | An integer containing only non-positive values (..,-2,-1,0) |
| positiveInteger | An integer containing only positive values (1,2,..) |
| short | A signed 16-bit integer |
| unsignedLong | An unsigned 64-bit integer |
| unsignedInt | An unsigned 32-bit integer |
| unsignedShort | An unsigned 16-bit integer |
| unsignedByte | An unsigned 8-bit integer |

También se puede definir si el elemento raiz puede tener un minimo o maximo de elementos hijos, entre muchas más restricciones posibles en este tipo de documentos,



## Evaluación de la solución

AL usar hojas de estilo xslt y ademas un schema con xsd, podemos estar seguros que nuestra aplicación no tendrá problemas al mostrarse como HTML o al comprobar errores de datos, ya que aplicando estos dos elementos hacemos nuestra solución más robusta comparada con una solución que solo cree y añada información a un documento XML.

## Aspectos positivos y negativos

Como he comentado en el apartado anterior, lo bueno de XML es que tiene comprobación de errores, tambien se le puede dar hojas de estilo, lo que permite darle formato a los datos, de esta forma se pueden obtener directamente de un fichero grande e imprimirlo en una pagina HTML, donde podrás ver todos los datos como lo desees, de esta forma serán fácilmente legibles y accesibles.

XML es una tecnología ampliamente utilizada, pero cada día hay nuevas tecnologías que la van reemplazando como pueden ser los JSON o bases de datos, las cuales son el numero 1 hoy en día para almacenar datos, son de mayor velocidad de consulta y tiene una gran cantidad de posibilidades de uso.