XML

Esta actividad trata de crear un XML con 3 elementos, uno de tipo string, uno de tipo numérico y el ultimo de tipo fecha, en mi caso utilicé el IDE Eclipse, y comencé por crear un nuevo archivo con el nombre de prueba.xml enseguida anexo el código del archivo:

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE usuario SYSTEM "test.dtd">

<usuario id="L-001">

<nombre>David Vazquez</nombre>

<edad>29</edad>

<fechaNac>1993-05-16</fechaNac>

</usuario>

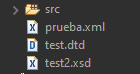
En estas líneas se explicará brevemente de que trata este archivo XML. Una vez estando en el IDE Eclipse, vamos a buscar en el menú de help>Eclipse Marketplace> y vamos a buscar e instalar Eclipse XML Editor and Tools, este plugin nos ayudara a validar si nuestro XML es válido y al igual nos ayudara a validarlo con DTD y con XML Scheme. Después de esto comenzamos a describir el código XML:

<?xml version="1.0" encoding="UTF-8"?>

La primera línea es la parte inicial del documento en donde definimos la versión del XML en donde indica que es un formato valido, seguido del tipo de codificación, es decir del tipo de caracteres que acepta.

<!DOCTYPE usuario SYSTEM "test.dtd">

La segunda línea es donde se indica y se hace referencia al archivo .dtd al cual se va a hacer referencia, y para esto tenemos que crear un archivo nombrado test.dtd dentro de la misma jerarquía en donde creamos nuestro archivo prueba.xml y quedara de la siguiente manera:



Una vez haciendo esto, el plugin empezará hacer su trabajo y nos empezará a enlazar el .xml y el .dtd y comenzará a marcar los errores, en ambos archivos.

En las siguientes líneas vamos a crear los elementos.

<usuario id="L-001">

<nombre>David Vazquez</nombre>

<edad>29</edad>

<fechaNac>1993-05-16</fechaNac>

</usuario>

En la primera línea definimos el elemento de usuario en donde le asigno un id como atributo, pensado que pudieran haber muchos usuarios podamos saber a cual nos referimos con ese id, dentro del elemento usuario creamos los elementos de nombre que contiene como valor un nombre en mi caso Davis Vazquez; el siguiente elemento es el de edad que va a contener un valor de 29, y el último elemento llamando fechaNac que contiene una fecha de nacimiento; con esta estructura cumplimos con lo solicitado que es tener un elemento de tipo String, uno de tipo numérico y el ultimo elemento de tipo fecha, cada elemento debe de contener su etiqueta de cierre.

<usuario id="L-001">

<nombre>David Vazquez</nombre>

<edad>29</edad>

<fechaNac>1993-05-16</fechaNac>

</usuario>

Con esto concluimos la creación del archivo prueba.xml, ahora vamos a editar el archivo test.dtd y enseguida agregare el código que debe de llevar el archivo:

<!ELEMENT usuario (nombre, edad, fechaNac)>

<!ELEMENT nombre (#PCDATA)>

<!ELEMENT edad (#PCDATA)>

<!ELEMENT fechaNac (#PCDATA)>

<!ATTLIST usuario id ID #REQUIRED>

Esta es la estructura que debe de llevar el archivo test.dtd, a continuación, explicaré la estructura del archivo.

<!ELEMENT usuario (nombre, edad, fechaNac)>

En la primera línea se comienza con la siguiente etiqueta <!ELEMENT> y sirve para definir un elemento que se encuentra en el archivo .xml (prueba.xml) y de la misma forma para los demás elementos se usa la misma etiqueta, ahora seguimos explicando la primera línea en donde después de definir el elemento agregamos el nombre del elemento que en este caso es usuario y como este elemento es padre de los demás elementos hijos, va fuera de los paréntesis y para los hijos los encerramos en paréntesis (elemento\_hijo\_1, elemento\_hijo\_2, elemento\_hijo\_3, etc.). Esta línea solo nos ayudará para identificar la estructura de los elementos de nuestro archivo prueba.xml.

<!ELEMENT nombre (#PCDATA)>

<!ELEMENT edad (#PCDATA)>

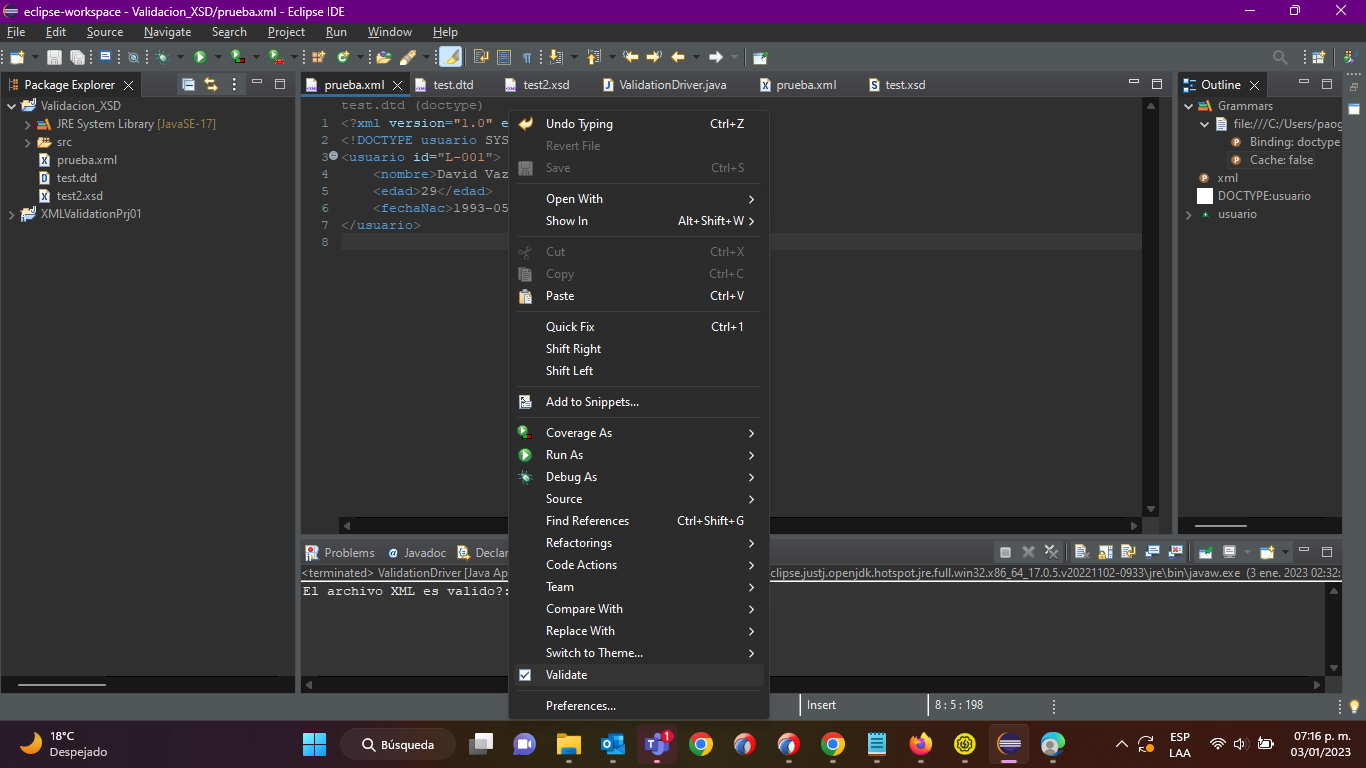
<!ELEMENT fechaNac (#PCDATA)>

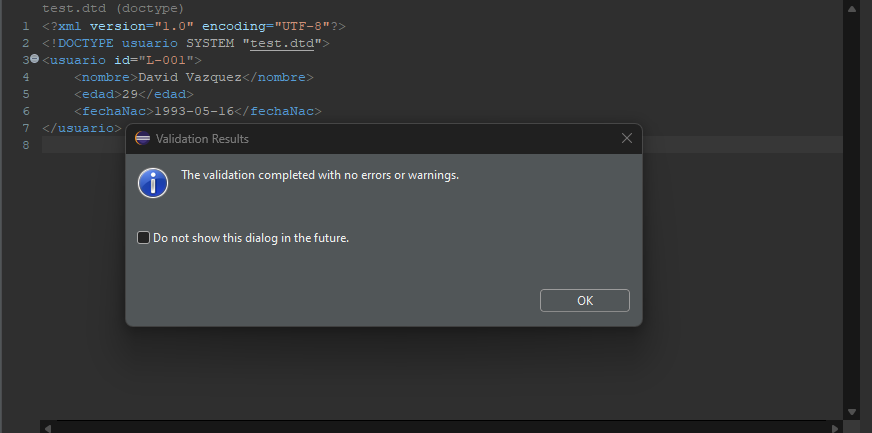
En estas líneas de igual manera definimos los elementos hijos, la cual son nombre, edad y fechaNac, una vez definido el elemento enseguida agregamos el nombre del elemento y lo que está dentro del paréntesis es la forma abreviada de decir el contenido que puede admitir el elemento. (#PCDATA) => (Parser Character Data), y así lo hacemos con cada uno de los elementos que tenemos definidos.

<!ATTLIST usuario id ID #REQUIRED>

Y para la última línea se agrega la etiqueta de <!ATTLIST> para definir que un elemento debe de contener un valor en este caso, al elemento usuario contiene un atributo id y se define que este sea del tipo de atributo de ID y con #REQUIRED se hace que el valor sea obligatorio.

Con esto concluimos con el archivo test.dtd, si se llegara a tener algún error en la creación del documento o en la edición, el IDE nos debe de marcar los errores, y para validar por último, hacemos clic derecho dentro de cualquier código de prueba.xml o test.dtd y buscamos casi al final la opción de validate y damos clic y nos mostrará si todo está en orden o en caso contrario nos indicará los errores.





Pasamos a la otra actividad en donde es validar el XML por medio de XML scheme, a continuación vamos a crear un documento más el cual será nombrado como test2.xsd y usaremos el mismo archivo prueba.xml solo le haremos algunas modificaciones, lo agrego enseguida:

<usuario xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:noNamespaceSchemaLocation="test2.xsd">

<nombre>David Vazquez</nombre>

<edad>29</edad>

<fechaNac>1993-05-16</fechaNac>

</usuario>

Las modificaciones que se le hicieron al archivo prueba.xml fueron las dos primeras líneas, para este caso solo queda así ya que la primera línea donde se define la versión y el tipo de caracteres admitido, porque en IDE Eclipse la marca como un error, solo puedes comentar y con eso es suficiente, la segunda línea es reemplazada por lo que agregamos en donde definimos el elemento usuario en donde definimos que el documento xml va a ser trabajado con XML Scheme, al igual que se basa en la normativa de la W3C Schema Definition Language xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance

Y en seguida agregamos xsi:noNamespaceSchemaLocation="test2.xsd" en donde hacemos agregamos un schemaLocation que hace referencia al archivo test.xsd, que será con el cual vamos a probar nuestro xml. De ahí en fuera es lo único que se modificó.

Ahora vamos a crear el archivo test.xsd y enseguida muestro el código:

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"

elementFormDefault="qualified">

<!-- Elementos simples-->

<xs:element name="nombre">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:maxLength value="30"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element name="edad">

<xs:simpleType>

<xs:restriction base="xs:int">

<xs:minExclusive value="0"/>

<xs:maxExclusive value="101"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element name="fechaNac">

<xs:simpleType>

<xs:restriction base="xs:date">

</xs:restriction>

</xs:simpleType>

</xs:element>

<!--elemento raiz-->

<xs:element name="usuario">

<xs:complexType>

<xs:sequence>

<xs:element ref="nombre" />

<xs:element ref="edad" />

<xs:element ref="fechaNac"/>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

En la primera línea se define le tipo de documento y de igual manera que la línea que se modificó que serán trabajados por XML Scheme y definimos el namespace que en este caso se refiere al schema y lleva por nombre xs, este xs debe de ir en todos los elementos de apertura y de cierre.

Ahora, enseguida del comentario donde dice Elementos simples, se comienzan a declarar los elementos que nuestro archivo prueba.xml contiene:

<xs:element name="nombre">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:maxLength value="30"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

De esta forma se definen los elementos, al abrir la etiqueta se pone el nombre del namespace en este caso xs, seguido de dos puntos y la palabra element, seguido del name de elemento. <xs:element name="nombre">

Como se mencionó anteriormente, para todas las etiquetas después de abiertas de debe anteponer en namespace seguido de los dos puntos, la palara siguiente simpleType sirve para definir un tipo simple y especifica las restricciones y la información sobre los valores de los atributos o elementos de solo texto. <xs:simpleType>

La siguiente línea es para agregar una restricción de que valor pueden ser admitidos por el elemento o atributo del XML, en este caso solo aceptaría valores de tipo String. <xs:restriction base="xs:string">

La siguiente línea define una longitud máxima de caracteres aceptables por el elemento en este caso son 30. <xs:maxLength value="30"/>

Y así con los demás elementos, en donde podemos ir agregando restricciones de acuerdo al tipo de dato que queremos que acepte nuestro archivo xml.

Ahora nos vamos a la parte donde está el comentario de elemento raíz, y ahí vamos a definir como está estructurado nuestros elementos en el xml, además en nuestro ejemplo vamos a agregar una secuencia de elementos. <xs:complexType> Agregamos la etiqueta de complexType para declarar que vamos a tener definir un elemento complejo y dentro de esta etiqueta vamos agregar una etiqueta de <xs:sequence> que nos va a indicar que va a tener una secuencia en la inserción de datos en los elementos de nuestro xml y por ultimo declaramos los elementos que se tienen en nuestro archivo xml, <xs:element ref="nombre" /> esta estructura se aplica para todos los elementos. Quedando de la siguiente manera:

<xs:element name="usuario">

<xs:complexType>

<xs:sequence>

<xs:element ref="nombre" />

<xs:element ref="edad" />

<xs:element ref="fechaNac"/>

</xs:sequence>

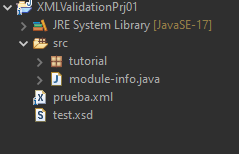
</xs:complexType>

</xs:element>

Y de igual manera el IDE Eclipse nos estará mostrando los errores en caso de que haya. Y listo de esta manera podemos validar nuestro archivo XML con DTD y con XML Schema.

Ahora la actividad que sigue es validar el XML dentro de un programa en Java, para esto vamos a usar los mismos archivos creados anteriormente, y lo único que se hará desde cero es el programa el cual dejo las instrucciones enseguida y el enlace del repositorio donde se encuentra el código fuente del programa.

Una vez estando en el IDE de Eclipse, vamos a crear una estructura como la de la siguiente imagen:



Esto con la finalidad de que sea lo más parecido a mi ejemplo y existan menos errores, en el archivo prueba.xml vamos a tener el siguiente código:

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<usuario>

<nombre>David Vazquez</nombre>

<edad>29</edad>

<fechaNac>1993-05-16</fechaNac>

</usuario>

Y en el archivo de test.sxd vamos a copiar la estructura del archivo tal cual lo tenemos en la actividad anterior donde fue creado, solo que este archivo ahora deberá de estar en la carpeta raíz de nuestro proyecto de java.

Este es el enlace del repositorio donde se encuentra el código.

[*https://github.com/dracko/XML.git*](https://github.com/dracko/XML.git)

El proyecto de java se llama XMLValidatorPrj01.

El proyecto de validación del XML con DTD y XML Scheme se llama Validacion\_XSD.

Y lo único que se tiene que hacer es modificar los nombres de los archivos tanto del XML como el de XSD si es que llegara a ser el caso.

Y podemos ejecutar y nos saldrá un mensaje en consola en donde indica si el archivo XML es válido o no, como lo muestro en la siguiente captura de pantalla.

Captura de pantalla de computadora

Descripción generada automáticamente