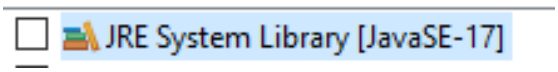
WATERML A HTML5

Esta aplicación se usa para convertir archivos WaterML de la versión 1.1 a HTML5. Los archivos de prueba que utilicé son los dos que hay metidos en la carpeta Ejercicio-3.

La extensión de WaterML para su versión 1.1 es .xml

El lenguaje que utilicé es Java en su versión 17.



El proyecto está organizado en dos paquetes: un paquete "file" en el que están las clases necesarias para leer y escribir ficheros y el paquete "main" en el que está la parte principal del código para convertir un WML a HTML5. Para ejecutar el programa, en la clase Main.java se crea una instancia del método WMLtoHTML5 hay que pasar como parámetros el nombre del archivo de entrada y el de salida.

WMLtoHTML5.java

Esta clase contiene varias clases en relación a cada elemento con los atributos que tienen en ellos: QueryInfo, SourceInfo, Coordenadas, LocalSite, Variable, Unit, Values, QualityControlLevel, Method, Source, ContractInformation, Sample y Laboratorio.

Su método principal *crearHTML5()* es el que es llamado por la clase Main y este llama en orden a los diferentes métodos que van creando el HTML5, llamando primero al método *inicio()* que escribe las primeras líneas del HTML5 como el <head>.

Después llama al método *parse()* que es fundamental. Primero crea una conexión al archivo XML usando la clase FileInputStream, después se obtiene una nueva instancia de un DocumentBuilderFactory para seguidamente conseguir una instancia de DocumentBuilder que usamos para poder conseguir un DOM (Document) de nuestro archivo XML. Tras esto, debemos crear una instancia de XPath que usaremos un poco más adelante.

```
FileInputStream XML = new FileInputStream("src/" + archivoXML);
DocumentBuilderFactory builderFactory = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = builderFactory.newDocumentBuilder();
Document xmlDocument = builder.parse(XML);
XPath xPath = XPathFactory.newInstance().newXPath();
```

Tras esto, debemos conseguir un NodeList usando el DOM que conseguimos antes y usando como expresión "//queryInfo" ya que es de ahí de dónde queremos sacar los datos.

```
NodeList nodelist0 = (NodeList)
xPath.compile("//queryInfo").evaluate(xmlDocument,
XPathConstants.NODESET);
```

Debemos recorrer el NodeList para poder ir sacando los datos cogiendo cada nodo individualmente. Todos los datos se sacan de la siguiente manera:

```
q.tiempoCreacion = xPath.compile("creationTime").evaluate(nodo0);
```

Como hay elementos dentro de otros como geogLocation dentro de geolocation debemos sacar sus nodos de otra manera:

```
NodeList nodelist2 = (NodeList)
xPath.compile("geoLocation").evaluate(nodo, XPathConstants.NODESET);
```

Y para conseguir los datos es de la misma forma que anteriormente, pero pasándole el nuevo nodo. Esto último ocurre muchas veces como qualityControlLevel, method, source etc. dentro de //values.

Lo siguiente es crear el cuerpo (<body>) del HTML5 añadiendo a una lista de cadenas los párrafos, títulos y demás según queramos.

Lo último que queda es llamar al método *writelines* pasando como parámetros el archivo HTML dónde queremos escribir la lista de cadenas que también pasamos como parámetro.

VALIDADORES

Todos los validadores pasan sin problemas. En el CSS saltan advertencias porque no hay color de primer plano y sí de fondo en algunos elementos, esto es porque se heredan al usar la cascada.