Trabajo CML



**Autores:** Rodrigo García y Sergio Bueno

Índice de Contenido

[Introducción 2](#_Toc533088432)

[XML 2](#_Toc533088433)

[CML 3](#_Toc533088434)

[API SAX 3](#_Toc533088435)

[Programa con API SAX 4](#_Toc533088436)

# **Introducción**

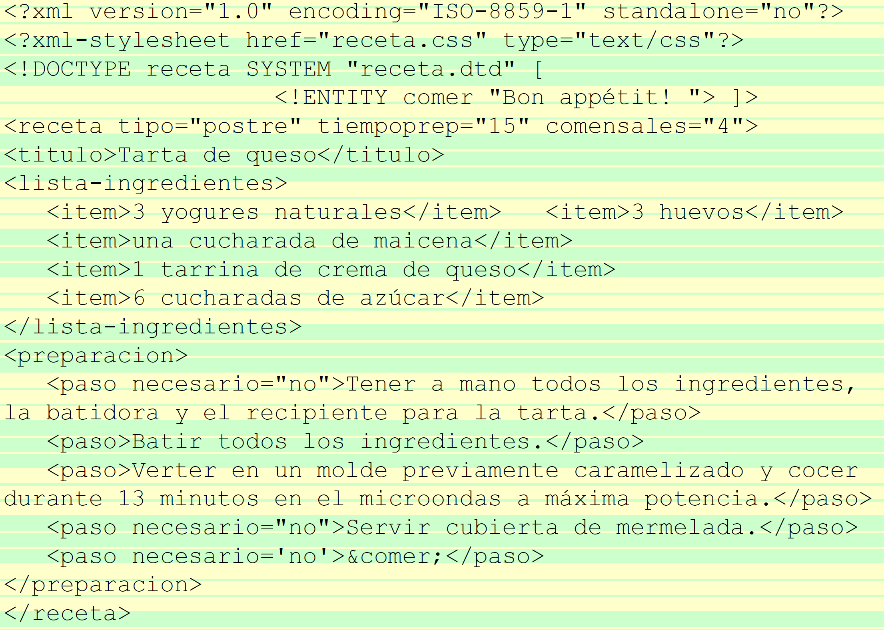
Durante el desarrollo de este trabajo vamos a implementar un lenguaje denominado CML (Chemical Markup Language), que es un tipo de lenguaje XML usado para expresar compuestos químicos, reacciones, cristales y química computacional.[[1]](#footnote-1)

Con este fin, antes de poder explicar el código requerido para esta práctica hemos creado, haremos una introducción a ambos lenguajes para poder entender correctamente ambos lenguajes. Además, haremos una breve introducción a la API SAX que hemos empleado en esta práctica y sus ventajas.

# XML

Es un simple, muy lenguaje de marcado derivado de SGML (Standard Generalized Markup Language ISO 8879: 1986). Diseñado originalmente para publicaciones electrónicas a gran escala, XML también se emplea para intercambiar una gran cantidad diferente de datos a través de internet.[[2]](#footnote-2)

A diferencia de otros lenguajes de marcado, XML no está predefinido por lo que podemos definir nuestras propias etiquetas (esto nos resulta muy útil puesto que nos permite emplear la API SAX previamente mencionada).

[[3]](#footnote-3)

En la imagen se puede observar el formato comienza siempre con una instrucción de procesado entre <? ... ¿>.

Luego está formado por elementos (ejemplo: <receta></receta>) que a su vez contiene diversos atributos (atributo = valor del atributo, ejemplo (tipo=”postre”)), de esta forma podemos definir a través de este lenguaje de marcado cualquier estructura que se nos ocurra. Cabe mencionar que la estructura interna de XML sería un árbol formado por nodos y hojas (elementos).

# CML

Chemical Markup Language es un lenguaje de marcado basado en XML pero en este caso está enfocado a la química, este lenguaje fue desarrollado por Peter Murray-Rust y Henry Rzepa desde 1995, es el lenguaje de facto XML para la química aceptado y publicado por todo el mundo. El problema con la química es que es imposible encontrar un modelo universal para la mayoría de los elementos, esto se debe a la diversidad de la química y por la cantidad de diferentes formas de químicos para organizar su información, también existen servicios de validación para un correcto uso del lenguaje.[[4]](#footnote-4)

# API SAX

SAX es la API simple para XML, fue la primera API de XML usada por todo el mundo y se convirtió en un estándar de facto. [[5]](#footnote-5) Permite interpretar un fichero web que usa XML, es mucho simple que su alternativa DOM (Document Object Model) es mucho más simple pero también dispone de menos capacidades para manipular el contenido de datos.

# Programa con API SAX

Está formado por varias clases y métodos. Definiremos los más importantes:

-XMLReader: Clase que implementa la interfaz SAX.

- CMLParser: Clase que implementa el analizador deseado. Emplear la API SAX nos permite analizar un documento XML pero no en un analizador que habrá que descargar de forma externa, en nuestro caso hemos empleado Apache Xerces <http://xerces.apache.org/xerces-j/> . En nuestro caso implementamos el parser con respecto a la gramática CML.

-CMLHandler: Clase que procesa el documento, el manejador de contenido y de léxico para nuestro fichero XML en formato CML. Una de las desventajas de SAX es que necesitamos almacenar en un objeto persistente, ya que no se dispone de estados a diferencia de DOM. Algunos de los métodos más importantes del mismo son:

• getRelativePath (): Método que obtiene el nivel de profundidad con respecto a la molécula en parseo.

• endDocument (): Método que finaliza el parseo del fichero y muestra sus resultados.

• printSubMolecules (Molecule molecule, int currentLevel): Método que muestra el árbol resultante del parseo actual.

• startElement (String namespaceURI, String localName, String qName, Attributes atts): Método que inicializa un Nuevo elemento

• endElement (String namespaceURI, String localName, String qName): Método que finaliza el parseo de un elemento y en caso de ser una molécula lo guarda en la lista de las mismas.

- CMLErrorHandler: Clase que gestiona el manejador de errores y warnings que puedan aparecer durante el parseo.

* warning (SAXParseException exception): Método que muestra un warning.
* error (SAXParseException exception): Método que muestra un error.
* fatalError (SAXParseException exception): Método que muestra un error fatal.

-XMLLocator: Clase que nos indica el nivel de profundidad de cada uno de los elementos parseados. Incluye los siguientes métodos:

* push (String element): Método que fija el nivel actual.
* pop (): Método que retrocede un nivel.
* getElement (): Método que devuelve el nivel actual.
* getLevel (): Método que devuelve la profundidad del nivel actual.
* getRoute (): Método que devuelve una lista de la ruta actual en orden ascendente.
* getPath (): Método que devuelve un String de la ruta actual en orden ascendente.

-Molecule: Clase que contiene toda la información acerca de una molécula. Contiene los siguientes métodos:

* Molecule (Map<String, Integer> atoms): Constructor de la clase molécula con un número previamente definido de átomos.
* setName (String name): Método que fija el nombre a la molécula.
* getName (): Método que muestra el nombre previamente fijado de la molécula.
* addElement (String element, int count): Método que añade un elemento a la molécula con un determinado número de átomos.
* addElement (String element): Método que añade un elemento a la molécula.
* addMolecule (Molecule molecule): Método para añadir una sub-molécula a una determinada molécula.
* getSubMolecules (): Método que devuelve una lista con todas las sub-moléculas de la molécula actual.
* getAllElements (): Método que devuelve una lista de átomos de la molécula actual.
* getElementMap (): Método que devuelve un mapa con un contador de todos los átomos de la molécula.
* getFormula (): Método que devuelve la fórmula química de la molécula actual.
* isOrganic (): Método que devuelve si una molécula es orgánica o no. En algunos casos puede dar un falso positivo ya que solo comprobamos que contenga Carbono.
* toString (): Método que nos devueve un String con los datos obtenidos de cada molécula.

1. Chemical Markup Language | CML [Internet]. Disponible en: http://www.xml-cml.org/ [↑](#footnote-ref-1)
2. Extensible Markup Language (XML) 1.0 (Fifth Edition) [Internet]. Disponible en: https://www.w3.org/TR/xml [↑](#footnote-ref-2)
3. «IntroXML4pdfPrint20130701.pdf». César I. G. Osorio. [↑](#footnote-ref-3)
4. Chemical Markup Language | CML [Internet]. Disponible en: http://www.xml-cml.org/documentation/history.html [↑](#footnote-ref-4)
5. SAX [Internet]. Disponible en: http://www.saxproject.org/ [↑](#footnote-ref-5)