# XML

**XML (Extensible Markup Language)** este un limbaj de marcare utilizat pentru a structura, stoca și transporta date într-un mod clar și ușor de înțeles atât de oameni, cât și de computere. XML este folosit frecvent în aplicații software și în schimbul de informații între sisteme.

XML organizează datele sub formă de elemente ierarhice (numite noduri) și atribute. Elementele sunt reprezentate prin taguri. Numele tagurilor pot fi alese de către dezvoltator (nu sunt predefinite, ca în alte limbaje de marcare precum HTML).

**Tipuri de tag-uri:**

* Tag de deschidere:

Marchează începutul unui element.

Are forma: <nume\_element>.

Exemplu: <model>.

* Tag de închidere:

Marchează sfârșitul unui element.

Are forma: <**/**nume\_element>.

Exemplu: <**/**model>.

* Tag-uri care se auto-închid:

Utilizate pentru elementele care nu conțin alte date sau elemente (deci pentru elemente vide).

Au forma: <fisier />.

Exemplu: <fisier nume="ceva.txt" />

Un **atribut** este o informație suplimentară atașată unui tag (element). Este utilizat pentru a descrie sau a adăuga proprietăți elementului, dar nu face parte din datele propriu-zise. Un atribut este definit întotdeauna în tag-ul de deschidere al unui element. Este format dintr-un nume și o valoare separate prin semnul egal =. Valoarea atributului este plasată între ghilimele duble drepte (") sau apostrofuri drepte ('). De exemplu în <fisier nume="ceva.txt" />, identificatorul nume este un atribut.

Primul rând al unui document XML este cunoscut sub numele de **declarație XML** și are rolul de a oferi informații despre versiunea XML utilizată și, opțional, despre codificarea caracterelor din document. Declarația are formatul <?xml version="[...]" encoding="[...]"?> unde veți înlocui [...] cu versiunea și codificarea corespunzătoare. De exemplu, versiunea poate fi 1.0 și codificarea poate fi UTF-8.

Pentru a verifica sintaxa unui fișier XML, puteți folosi un validator XML: <https://jsonformatter.org/xml-validator>

Exemplu de XML:

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

<obiecte>

<obiect id="100" nume="zar">

<model info="cub">1</model>

<textura folder="texturi\_zar">7</textura>

</obiect>

<obiect id="27" nume="minge">

<model info="sfera">3</model>

<textura folder="texturi\_jucarii">23</textura>

</obiect>

</obiecte>

Structura unui XML este similară cu a unui **arbore**. Într-un arbore XML:

* Rădăcina este primul element al documentului și conține toate celelalte elemente. Un document XML valid trebuie să aibă un singur element rădăcină.
* Fiecare element din XML poate avea sub-elemente numite noduri-copil. Un element care conține alte elemente se numește nod-părinte.

De exemplu, pentru:

<lista\_modele>

<model id="1">

<model id="2">

</lista\_modele>

Elementele model sunt copii pentru elementul lista\_modele, iar lista\_modele este părinte pentru elementele model.

* Nodurile fără alte sub-noduri sau conținut sunt numite noduri-frunze.

Structura de arbore pentru XML-ul dat mai sus este:

obiecte

├── obiect (id="100", nume="zar")

│ ├── model (info="cub"): "1"

│ └── textura (folder="texturi\_zar"): "7"

└── obiect (id="27", nume="minge")

├── model (info="sfera"): "3"

└── textura (folder="texturi\_jucarii"): "23"

Deoarece fișierele de configurare pentru GameEngine vor fi în XML, vom folosi o bibliotecă pentru parsarea acestui tip de fișiere. Ca exemplu, am ales RapidXML, însă dacă nu vă place, puteți să alegeți, evident, o altă bibliotecă pentru proiect. De asemenea, daca nu doriți să salvați datele în XML, puteți alege alt format, cum ar fi JSON. Puteți downloada RapidXML de aici: [http://rapidxml.sourceforge.net/](https://rapidxml.sourceforge.net/?fbclid=IwAR304BJ_L3wqmsUjswUzp_DGMi7mY3tuoiyJJiODSY54tOP_0kFEJx-XDMA)

## RapidXML

RapidXML este o bibliotecă foarte rapidă pentru procesarea fișierelor XML în C++. Este o bibliotecă header-only (doar fișiere .hpp), ceea ce înseamnă că trebuie doar să includem fișierele de antet în proiect pentru a o folosi.

Pentru a citi fișierele XML avem nevoie să includem:

#include "rapidxml.hpp"

#include "rapidxml\_utils.hpp"

**Atenție headerele de rapidXML, sau headerele care includ rapidxml trebuie să fie înainatea include-ului**:

#include "../Utilities/utilities.h"

RapidXML folosește namespace-ul rapidxml. Pentru a nu avea confuzii de nume, ar fi bine să evităm de data asta instrucțiunea using namespace rapidxml.

RapidXML nu citește direct fișiere, ci operează pe un buffer de memorie. Putem folosi rapidxml::file<> pentru a încărca un fișier XML într-un buffer.

rapidxml::file<> file("exemplu.xml");

char\* buffer = file.data();

Pentru a parsa fișierul, vom creea un obiect de tip rapidxml::xml\_document<> și vom folosi metoda parse:

rapidxml::xml\_document<> doc;

doc.parse<0>(buffer);

Pentru a declara o varaibilă de tip nod din arborele XML rezultat în urma parsării, folosim tipul de date:

rapidxml::xml\_node<>\* nod;

Pentru a obține nodul rădăcină al documentului:

rapidxml::xml\_node<>\* root = doc.first\_node("root");

if (!root) {

std::cerr << "Nu s-a găsit nodul rădăcină!" << std::endl;

return 1;

}

Pentru a parcurge nodurile-copil din arborele XML pentru un nod-parinte, putem folosi o secvență de cod asemănătoare cu următoarea:

for (rapidxml::xml\_node<>\* node = nod\_parinte->first\_node(); node; node = node->next\_sibling()) {

// ... proceseaza nodurile ...

}

Pentru a accesa primul fiu al unui nod-părinte folosim:

nod\_parinte->first\_node();

Atunci când dorim să ne duce la un nod-copil specific (un nod al cărui nume îl cunoaștem) putem specificanumele acestuia ca parametru în metoda first\_node()

nod\_parinte->first\_node("nume\_tag")

Putem să ne deplasăm la următorul nod "frate" al unui nod cu

nod = nod->next\_sibling();

Sauputem chiar să specificăm și numele următorului nod frate la care să ne deplasăm (va sări peste frații cu nume diferite)

nod = nod->next\_sibling("nume\_nod");

Pentru a obține numele unui nod (sub formă de array de char):

node->name()

Pentru a afișa valoarea unui nod:

node->value()

Pentru a obține primul atribut (indiferent de nume):

node->first\_attribute()

Putem specifica și numele atributului dorit:

node->first\_attribute("nume\_atribut")

Putem itera print atributele unui nod cu un cod de genul:

std::cout << "Atributele nodului '" << node->name() << "':" << std::endl;

for (rapidxml::xml\_attribute<>\* attr = node->first\_attribute(); attr; attr = attr->next\_attribute()) {

std::cout << " - " << attr->name() << " = " << attr->value() << std::endl;

}

Observăm că și atributele au metodele name() și value() care returnează câte un array de caractere cu numele și respectiv valoarea lor.

Pentru a trece la următorul atribut al nodului s-a folosit:

attr->next\_attribute()

## Bibliografie

1. Un tutorial micut de RapidXML:

[https://semidtor.wordpress.com/.../rapidxml-mini-tutorial/](https://semidtor.wordpress.com/2013/03/29/rapidxml-mini-tutorial/?fbclid=IwAR1R1yAVJCvIewdlEYyTfRUYdp3tLSiBCBXvyhknCSgmiIBTSETTwfuiz5I)

1. Manualul oficial de RapidXML:

<https://rapidxml.sourceforge.net/manual.html>

## Task

Ca sa va antrenati sa lucrati cu aceasta biblioteca, aveti si un fisier micut de tip XML la adresa:<https://drive.google.com/drive/folders/0B2T1XQJjbPctN1IyUTZNT2RZYW8?resourcekey=0-mkuYuptDnHTSY710XPXwVA>

pe care o sa va rog sa-l parsati cu ajutorul lui RapidXML. Puteti lua si un xml oarecare de pe net sau chiar sa-l faceti voi. Important e sa invatati sa afisati continutul fisierului xml, trecand prin fiecare element in parte - si e mult mai simplu sa faceti asta pornind de la un fisier micut decat de la fisierul xml de configurare pentru game engine.

De exemplu, un mod de afisare pentru un tag oarecare ar putea fi:

nume\_element ‍‍‍‍‍‍ ‍‍cu atributele atr1=val1, atr2=val2

cu continutul ...‍‍‍‍‍‍ ‍‍

cu fiii:

nume\_alt\_element etc

Concret, pentru xml-ul dat ca exemplu:

obiecte

fara atribute

‍‍‍‍‍‍ ‍‍cu fiii:

‍‍‍‍‍‍ ‍‍ ‍‍‍‍‍‍ ‍‍obiect

‍‍‍‍‍‍ ‍‍ ‍‍‍‍‍‍ ‍‍ ‍‍‍‍‍‍ ‍‍cu atributele id=100, nume=zar

‍‍‍‍‍‍ ‍‍ ‍‍‍‍‍‍ ‍‍ ‍‍‍‍‍‍ ‍‍cu fiii:

‍‍‍‍‍‍ ‍‍ ‍‍‍‍‍‍ ‍‍ ‍‍‍‍‍‍ ‍‍ ‍‍‍‍‍‍ ‍‍model

cu atributele info=cub

‍‍cu continutul 1 (aici nu avem fii) .....etc.

Daca nu va place modul acesta de reprezentare, puteti face un altul, ideea e sa stiti sa parcurgeti fiii, atributele si continutul unui element, si, in general, sa va puteti "deplasa" fara probleme prin structura unui XML.