Esercizio sull'eXtensible Markup Language

Vincenzo Turturro

1/11/2020

Laboratorio TPS

INTRODUZIONE

Nel lontano 1998, il W3C (World Wide Web Consortium), in seguito alla creazione da parte di diverse aziende (Microsoft, Netscape, etc.) di diverse estensioni proprietarie dell'HTML, decise di creare un linguaggio di markup che desse maggiore liberta' nella definizione dei tag, pur dettando degli standard.

Dall'ora, tale metalinguaggio ricevette sempre piu' attenzione e si diffuse fino a diventare oggi una base per la definizione di altri linguaggio o applicazioni, come ad esempio successo con le immagini vettoriali (le quali infatti sono un'estensione dell'XML).

LA STRUTTURA

Come gia' detto, la caratteristica dell'XML e' quella di permettere al programmatore di definire senza restrizioni diversi tag tra loro diversi. Nonostante cio', affinche' la struttura sia valida si sono introdotti nel tempo una serie di standard. Il documento deve infatti iniziare con il seguente tag:

```
<?xml version="1.0"?>
```

e la struttura del documento xml deve essere gerarchica ovvero una struttura nella quale vi e' una relazione tra "figlio e padre" (attuata attraverso una struttura di tipo albero). Tale relazione e' possibile grazie all'elemento, il blocco base del documento XML che viene delimitato da un tag di apertura ed un tag di chiusura.

```
<greeting>Hello, world!</greeting>
```

Ogni elemento puo' inoltre, al suo interno contenere altri elementi (da qui la struttura gerarchica). Infine ad ogni elemento possono essere associati gli attributi il cui compito e' quello di fornire informazioni addizionali all'elemento.

```
<LIBRO RILEGATURA="brossura">Decameron</LIBRO>
```

Oltre agli elementi e gli attributi, un documento xml puo' essere composto da entita' ma, queste non verranno approfondite in tale momento.

STUDY CASE

Prendiamo come study case quello di una libreria che ha la necessita' di catalogare i

libri. Dopo diverse ricerche, si sceglie come modello di catalogazione il modello dewey (utilizzato anche dall'opac) secondo il quale i libri sono classificati attraverso 10 classi (ad ognuna delle quali e' associato un numero):

- 000-099 Generalità;
- 100-199 Filosofia e Psicologia;
- 200-299 Religione;
- 300-399 Scienze Sociali;
- 400-499 Linguaggio;
- 500-599 Scienze Naturali;
- 600-699 Tecnologia e scienze applicate;
- 700-799 Arti;
- 800-899 Letteratura e Retorica;
- 900-999 Geografia e Storia.

Si procede quindi studiando le caratteristiche basi di un libro e ipotizzando la struttura base di questo:

```
<book>
    <isbn></isbn>
    <title></title>
    <ean></ean>
    <price></price>
    <authors>
        <author></author>
        <author></author>
    </authors>
    <publisher></publisher>
    <audience_age></audience_age>
    <release_date></release_date>
    <studio></studio>
    <pages></pages>
    <weight></weight>
    <rating></rating>
```

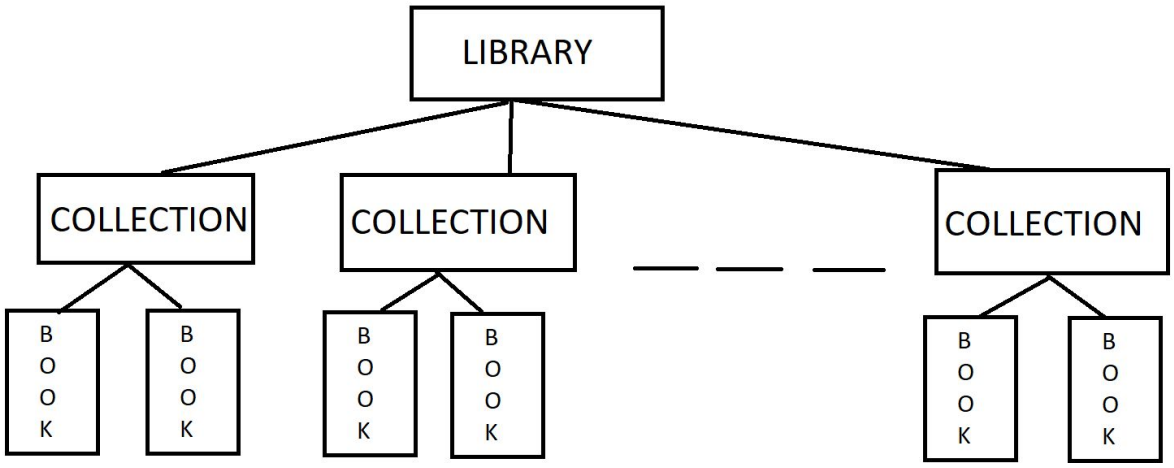
```
            <genres>
                <genre></genre>
                <genre></genre>
            </genres>
            <language></language>
            <country></country>
            <summary> </summary>
        </book>
```

Successivamente, si passa quindi ad implementare il singolo libro in una struttura gerarchica piu' estesa:

```
<?xml version="1.0" encoding="UTF-8" ?>
<library>
    <collection dewhy="000">
      <book>
      </book>
      <book>
      </book>
    </collection>
    <collection dewhy="100">
      <book>
      </book>
      <book>
      </book>
    </collection>
```

```
    <collection dewhy="900">
      <book>
      </book>
      <book>
      </book>
    </collection>
</library>
```

La struttura che ne esce e' quindi la seguente:



dove la libreria e' l'insieme delle collezioni (fisicamente gli scaffali) le quale raggruppano i libri secondo la prima cifra decimale del codice dewey. Viene inoltre aggiunto come attributo nel libro il codice dewey completo.

ESEMPIO DI CATALOGAZIONE

Successivamente, si riporta un esempio completo della struttura xml sopra' definita:

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

<library>

    <collection dewhy="000">

        <book full_dewhy="015">

            <isbn>9788850318483</isbn>

            <title>Java 11: Guida allo sviluppo in ambienti Windows, macOS
e GNU/Linux</title>

            <ean>883929169931</ean>

            <price>34.99</price>

            <authors>

                <author>Apogeo</author>

            </authors>

            <publisher>Apogeo Editore</publisher>

            <audience_age>16+</audience_age>

            <release_date>8/12/2015</release_date>

            <studio>Apogeo Editore</studio>

            <pages>302</pages>

            <weight>453</weight>

            <rating>4.1</rating>

            <genres>
```

```
            <genre>Computer</genre>
            <genre>Programmazione</genre>
        </genres>
        <language>italian</language>
        <country>Italy</country>
        <summary> Java is a class-based, object-oriented programming
language that is designed to have as few implementation dependencies as
possible. It is a general-purpose programming language intended to let
application developers write once, run anywhere (WORA), meaning that
compiled Java code can run on all platforms that support Java without the
need for recompilation. </summary>
    </book>
    <book full_dewhy="016">
        <isbn>9788850318261</isbn>
        <title>Clean Architecture: Guida per diventare abili
progettisti di architetture software</title>
        <ean>883929169931</ean>
        <price>24.99</price>
        <authors>
            <author>Apogeo</author>
        </authors>
        <publisher>Apogeo Editore</publisher>
        <audience_age>16+</audience_age>
        <release_date>16/04/2018</release_date>
        <studio>Apogeo Editore</studio>
        <pages>302</pages>
```

```
            <weight>453</weight>
            <rating>4.9</rating>
            <genres>
                <genre>Computer</genre>
                <genre>Programmazione</genre>
            </genres>
            <language>italian</language>
            <country>Italy</country>
            <summary> By applying universal rules of software
architecture, you can dramatically improve developer productivity
throughout the life of any software system. Now, building upon the success
of his best-selling books Clean Code and The Clean Coder, legendary
software craftsman Robert C. Martin . </summary>
        </book>
    </collection>
    <collection dewhy="100">
        <book full_dewhy="170">
            <isbn>9788854124592</isbn>
            <title>L'interpretazione dei sogni</title>
            <ean>883929169931</ean>
            <price>4.99</price>
            <authors>
                <author>Sigmund Freud</author>
            </authors>
            <publisher>Newton Compton Editor</publisher>
```

```
            <audience_age>18+</audience_age>

            <release_date>5/11/2010</release_date>

            <studio>Newton Compton Editor</studio>

            <pages>302</pages>

            <weight>453</weight>

            <rating>3.9</rating>

            <genres>

                <genre>Psicologia</genre>

                <genre>Filosofia</genre>

                <genre>Etica</genre>

            </genres>

            <language>italian</language>

            <country>Italy</country>

            <summary>Lorem ipsum dolor sit amet, consectetur adipiscing
elit. In dignissim ac nisi ac dapibus. In hac habitasse platea dictumst.
Proin scelerisque justo eget tempor accumsan. Nulla non pellentesque sem,
vitae auctor massa. Maecenas consectetur dolor et lectus blandit, sit amet
condimentum ante efficitur. Curabitur et consequat velit. Fusce tincidunt
ante nibh, ut porta est interdum at. Ut pulvinar ultricies posuere.
Curabitur a lobortis felis, vitae commodo risus. </summary>

        </book>

    </collection>

    <collection dewhy="200">

        <book full_dewhy="290">

            <isbn>9788867084333</isbn>

            <title>Buddhismo. Religione senza religione</title>
```

```
            <ean>883929169931</ean>
            <price>21.99</price>
            <authors>
                <author>Alan Watts</author>
            </authors>
            <publisher>Edizioni Lindau</publisher>
            <audience_age>18+</audience_age>
            <release_date>5/11/2010</release_date>
            <studio>Edizioni Lindau</studio>
            <pages>109</pages>
            <weight>109</weight>
            <rating>5</rating>
            <genres>
                <genre>Religione</genre>
                <genre>Filosofia</genre>
            </genres>
            <language>italian</language>
            <country>Italy</country>
            <summary> Lorem ipsum dolor sit amet, consectetur adipiscing
elit. In dignissim ac nisi ac dapibus. In hac habitasse platea dictumst.
Proin scelerisque justo eget tempor accumsan. Nulla non pellentesque sem,
vitae auctor massa. Maecenas consectetur dolor et lectus blandit, sit amet
condimentum ante efficitur. Curabitur et consequat velit. Fusce tincidunt
ante nibh, ut porta est interdum at. Ut pulvinar ultricies posuere.
Curabitur a lobortis felis, vitae commodo risus.</summary>
        </book>
```

```
    </collection>
    <collection dewhy="300">
        <book full_dewhy="321">
            <isbn>9788899301453</isbn>
            <title>Politica italiana e Nuovo Ordine Mondiale</title>
            <ean>883929169931</ean>
            <price>40.99</price>
            <authors>
                <author>Gabriele Sannino</author>
            </authors>
            <publisher>Fuoco Edizioni</publisher>
            <audience_age>18+</audience_age>
            <release_date>5/11/2010</release_date>
            <studio>Fuoco Edizioni</studio>
            <pages>254</pages>
            <weight>179</weight>
            <rating>3.4</rating>
            <genres>
                <genre>Politica</genre>
            </genres>
            <language>italian</language>
            <country>Italy</country>
            <summary> Lorem ipsum dolor sit amet, consectetur adipiscing
elit. In dignissim ac nisi ac dapibus. In hac habitasse platea dictumst.
```

```
Proin scelerisque justo eget tempor accumsan. Nulla non pellentesque sem,
vitae auctor massa. Maecenas consectetur dolor et lectus blandit, sit amet
condimentum ante efficitur. Curabitur et consequat velit. Fusce tincidunt
ante nibh, ut porta est interdum at. Ut pulvinar ultricies posuere.
Curabitur a lobortis felis, vitae commodo risus.</summary>

        </book>

    </collection>

    <collection dewhy="400">

        <book full_dewhy="494">

            <isbn>9781462901036</isbn>

            <title>Korean for Dummies</title>

            <ean>883929169931</ean>

            <price>10.99</price>

            <authors>

                <author>Korean Nunny</author>

            </authors>

            <publisher>Tuttle Publishing</publisher>

            <audience_age>13+</audience_age>

            <release_date>10/05/2010</release_date>

            <studio>Tuttle Publishing</studio>

            <pages>176</pages>

            <weight>179</weight>

            <rating>4.4</rating>

            <genres>

                <genre>Lingua</genre>
```

```
            </genres>
            <language>italian</language>
            <country>Italy</country>
            <summary>Lorem ipsum dolor sit amet, consectetur adipiscing
elit. In dignissim ac nisi ac dapibus. In hac habitasse platea dictumst.
Proin scelerisque justo eget tempor accumsan. Nulla non pellentesque sem,
vitae auctor massa. Maecenas consectetur dolor et lectus blandit, sit amet
condimentum ante efficitur. Curabitur et consequat velit. Fusce tincidunt
ante nibh, ut porta est interdum at. Ut pulvinar ultricies posuere.
Curabitur a lobortis felis, vitae commodo risus. </summary>
        </book>
    </collection>
    <collection dewhy="500">
        <book full_dewhy="511">
            <isbn>9788865378304</isbn>
            <title>Sette lezioni di fisica</title>
            <ean>883929169931</ean>
            <price>20.92</price>
            <authors>
                <author>Leo Sgulli</author>
            </authors>
            <publisher>Edizioni del Faro</publisher>
            <audience_age>18+</audience_age>
            <release_date>10/05/2010</release_date>
            <studio>Edizioni del Faro</studio>
            <pages>267</pages>
```

```
            <weight>179</weight>
            <rating>4.4</rating>
            <genres>
                <genre>Scienza</genre>
                <genre>Fisica</genre>
            </genres>
            <language>italian</language>
            <country>Italy</country>
            <summary> Lorem ipsum dolor sit amet, consectetur adipiscing
elit. In dignissim ac nisi ac dapibus. In hac habitasse platea dictumst.
Proin scelerisque justo eget tempor accumsan. Nulla non pellentesque sem,
vitae auctor massa. Maecenas consectetur dolor et lectus blandit, sit amet
condimentum ante efficitur. Curabitur et consequat velit. Fusce tincidunt
ante nibh, ut porta est interdum at. Ut pulvinar ultricies posuere.
Curabitur a lobortis felis, vitae commodo risus.</summary>
        </book>
    </collection>
    <!--<collection dewhy="600">
        useless since we have no child
    </collection>-->
    <!--<collection dewhy="700">
        useless since we have no child
    </collection>-->
    <collection dewhy="800">
        <book full_dewhy="851">
            <isbn>9781781101582</isbn>
```

```
            <title>Harry Potter and the Philosopher Stone</title>

            <ean>883929169931</ean>

            <price>8.99</price>

            <authors>

                <author>J.K. Rowling</author>

            </authors>

            <publisher>Pottermore Publishing</publisher>

            <audience_age> 9-18 </audience_age>

            <release_date>8/12/2015</release_date>

            <studio>Pottermore Publishing</studio>

            <pages>302</pages>

            <weight>148</weight>

            <rating>4.8</rating>

            <genres>

                <genre>Action</genre>

                <genre>Fantasy</genre>

            </genres>

            <language>italian</language>

            <country> Italy </country>

            <summary> Harry Potter has been living a difficult life,
constantly abused by his surly and cold aunt and uncle, Vernon and Petunia
Dursley and bullied by their spoiled son Dudley since the death of his
parents ten years prior. His life changes on the day of his eleventh
birthday when he receives a letter of acceptance into Hogwarts School of
Witchcraft and Wizardry, delivered by a half-giant named Rubeus Hagrid
after previous letters had been destroyed by Vernon and Petunia. Hagrid
```

```
details Harry's past as the son of James and Lily Potter, who were a
wizard and witch respectively, and how they were murdered by the most evil
and powerful dark wizard of all time, Lord Voldemort, which resulted in
the one-year-old Harry being sent to live with his aunt and uncle.
Voldemort was not only unable to kill Harry, but his powers were also
destroyed in the attempt, forcing him into exile and sparking Harry's
immense fame among the magical community.</summary>

        </book>

        <book full_dewhy="831">

            <isbn>9788858630020</isbn>

            <title>The Picture Of Dorian Gray</title>

            <ean>883929169931</ean>

            <price>8.99</price>

            <authors>

                <author>Oscar Wilde</author>

            </authors>

            <publisher>Blur</publisher>

            <audience_age> 18-40 </audience_age>

            <release_date>22/05/2012</release_date>

            <studio>Blur</studio>

            <pages>256</pages>

            <weight>121</weight>

            <rating>4.3</rating>

            <genres>

                <genre>Classic</genre>

            </genres>
```

```
            <language>Inglese</language>

            <country> Italy </country>

            <summary> The Picture of Dorian Gray begins on a beautiful
summer day in Victorian England, where Lord Henry Wotton, an opinionated
man, is observing the sensitive artist Basil Hallward painting the
portrait of Dorian Gray, a handsome young man who is Basil's ultimate
muse. While sitting for the painting, Dorian listens to Lord Henry
espousing his hedonistic world view and begins to think that beauty is the
only aspect of life worth pursuing, prompting Dorian to wish that his
portrait would age instead of himself.</summary>

        </book>

        <book full_dewhy="831">

            <isbn>9788858630020</isbn>

            <title>La ragazza dagli occhi di carta</title>

            <ean>883929169931</ean>

            <price>8.99</price>

            <authors>

                <author>Ilaria Tuti</author>

            </authors>

            <publisher>Associazione Culturale Nero Cafè</publisher>

            <audience_age> 18-40 </audience_age>

            <release_date>22/05/2012</release_date>

            <studio>Associazione Culturale Nero Cafè</studio>

            <pages>223</pages>

            <weight>121</weight>

            <rating>2.3</rating>
```

```
            <genres>
                <genre>Classic</genre>
                <genre>Giallo</genre>
            </genres>
            <language>italian</language>
            <country> Italy </country>
            <summary>Lorem ipsum dolor sit amet, consectetur adipiscing
elit. In dignissim ac nisi ac dapibus. In hac habitasse platea dictumst.
Proin scelerisque justo eget tempor accumsan. Nulla non pellentesque sem,
vitae auctor massa. Maecenas consectetur dolor et lectus blandit, sit amet
condimentum ante efficitur. Curabitur et consequat velit. Fusce tincidunt
ante nibh, ut porta est interdum at. Ut pulvinar ultricies posuere.
Curabitur a lobortis felis, vitae commodo risus.</summary>
        </book>
    </collection>
    <collection dewhy="900">
        <book full_dewhy="955">
            <isbn>9788858429969</isbn>
            <title>Dall'Asia al mondo</title>
            <ean>883929169931</ean>
            <price>20.92</price>
            <authors>
                <author>Pierre Grosser</author>
            </authors>
            <publisher>Giulio Einaudi Editore</publisher>
            <audience_age>18+</audience_age>
```

```
            <release_date>10/05/2010</release_date>
            <studio>Giulio Einaudi Editore</studio>
            <pages>267</pages>
            <weight>179</weight>
            <rating>4.4</rating>
            <genres>
                <genre>Storia</genre>
                <genre>Geografia</genre>
            </genres>
            <language>italian</language>
            <country>Italy</country>
            <summary> Lorem ipsum dolor sit amet, consectetur adipiscing
elit. In dignissim ac nisi ac dapibus. In hac habitasse platea dictumst.
Proin scelerisque justo eget tempor accumsan. Nulla non pellentesque sem,
vitae auctor massa. Maecenas consectetur dolor et lectus blandit, sit amet
condimentum ante efficitur. Curabitur et consequat velit. Fusce tincidunt
ante nibh, ut porta est interdum at. Ut pulvinar ultricies posuere.
Curabitur a lobortis felis, vitae commodo risus.</summary>
        </book>
    </collection>
</library>
```

ESEMPIO DI UTILIZZO

Possiamo utilizzare la struttura xml sopra descritta per creare una piccola applicazione client-server nel quale il server invia i dati sotto formato xml al client il quale si occupa invece di visualizzarli sullo schermo.

COSA E' UN SOCKET?

Un socket è una struttura software astratta che consente la comunicazione (trasferimento di dati) tra due processi su internet indipendente dall'interfaccia di rete. Nel gergo socket uno dei processi che comunicano è chiamato Server (il receiver) e l'altro Client (il sender). Tra i due processi il server è quello che ha controllo maggiore, poiché è il processo che accetta o meno la connessione. Il fatto che un programma agisca come client o come server determina un differente uso delle API che il linguaggio fornisce (in questo caso Java)

1. Il Client(il sender) ha bisogno di conoscere l'indirizzo del server ed utilizza la classe socket per effettuare una richiesta di connessione al server.
2. Il Server (il receiver) può apprendere informazioni sull'indirizzo del client una volta stabilita la connessione ed utilizza la classe serverSocket per rimanere in ascolto su una porta.

Quindi le classi utilizzate sono :

1. serverSocket per metterci in ascolto
2. socket per connetterci al server e contenere le informazioni relative alla connessione quali lo stream di input e quello di output
3. InetAddress, un insieme di metodi statici per operare con gli indirizzi di rete

APP LATO SERVER

Si passa quindi alla stesura dell'app lato server il cui compito sara' quello di inviare i dati (sotto forma di stringa xml) al client.

Per prima cosa, scriviamo la funzione che leggera' i dati e li memorizzera' in una stringa

:

```
    private String read(){
        try{
            File iFile = new File("books.xml");
            Scanner sc = new Scanner(iFile);

            StringBuilder sb = new StringBuilder();

            while(sc.hasNextLine()){
                sb.append(sc.nextLine());
            }

            return sb.toString();
        }catch (Exception e ){
            e.printStackTrace();
        }

        return "";
    }
```

Utilizziamo un oggetto Scanner per accedere allo stream del file e un oggetto StringBuilder per non creare accumuli di memoria inutilizzati.

Successivamente, startiamo il server su una porta qualsiasi (in questo caso 7777) e, ci mettiamo in ascolto di richieste di connessioni. Utilizziamo come protocollo di trasporto

il TCP.

```
public class Server {

    private ServerSocket serverSocket;

    public Server(){

    }

    public void start(int port){

        try {

            serverSocket = new ServerSocket(port);

            System.out.println("[INFO] Server started ");

        } catch (Exception e) {

            System.exit(-1);

        }

        while(true){

            try {

                new ClientHandler(serverSocket.accept()).start();

            } catch (Exception e) {

                System.err.println();

            }

        }

    }

}
```

Andiamo infine a scrivere la classe che si occupera' di gestire la connessione col client e quindi di inviare i dati:

```
public class ClientHandler extends Thread{

    private final Socket clientSocket;

    private PrintWriter out;

    private BufferedReader in;

    public ClientHandler(Socket socket){

        this.clientSocket = socket;

        try {

            out = new PrintWriter(clientSocket.getOutputStream(), true);

            in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));

        }catch (Exception e){

            e.printStackTrace();

        }

    }

    public void run(){

        try {

            String xmlFile = read();

            out.write(xmlFile);

            out.flush();

            in.close();

            out.close();

            clientSocket.close();
```

```
        }catch (Exception ignored){}

    }



    private String read(){//riportato sopra}

}
```

Nel nostro caso, le operazioni sulla struttura xml verranno svolte da lato client per comodita'.

LA MANIPOLAZIONE XML IN JAVA

Java e XML sono due tecnologie che si completano vicendevolmente:

1. Da un lato XML fornisce una sintassi standard e uniforme per la rappresentazione dei dati. Questo consente agli sviluppatori di produrre un codice portabile che si interfaccia con dati rappresentati in una maniera altrettanto portabile.

2. D'altro lato Java fornisce grazie al supporto nativo per la manipolazione di stringhe, l'ambiente ideale per sviluppare delle API che consentano di manipolare l'XML.

L'api per la manipolazione dell'xml in java (JAXP) si compone di tre librerie. Di queste andremo ad utilizzare DOM (Document object modifier) la quale ci permettera' di trasformare un documento xml in un albero di oggetti che riflette la struttura del documento. L'albero può poi essere manipolato e riconvertito in un nuovo documento XML.

Le classi che compongono la libreria DOM sono:

1. DocumentBuilderFactory e DocumentBuilder le quale ci permettono di ottenere a partire da un documento .xml, una struttura ad albero sul quale poter operare;
2. Document che rappresenta per l'appunto l'albero;
3. NodeList, una raccolta ordinata di nodi;
4. Node, il tipo di base primario per l'intero albero
5. Element, i nodi base dell'albero

LETTURA DATI XML

```
public void fun(){

    try{

        File xmlFile = new File("path_file");

        DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();

        DocumentBuilder dBuilder = factory.newDocumentBuilder();

        Document doc = dBuilder.parse(xmlFile);

        doc.getDocumentElement().normalize();

        //doc.getDocumentElement().getNodeName(); per l'elemnento root

        NodeList nodeList = doc.getElementsByTagName("nodi_che_vogliamo");

        for(int x=0; x<nodeList.getLenght(); x++){

            Node child = nodeList.item(x);

            //verifico che il nodo sia un elemento

            if(child.getNodeType == Node.ELEMENT_NODE){

                Element element = (Element) child;

                //effettuo le operazioni sull'elemento

                element.getElementsByTagName("tag");

                element.getAttribute("attrib");

                //etc

            }

        }

    }catch(Exception ignored){}
```

```
}
```

SCRITTURA DATI XML

```
public void fun(){

    try{

        DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();

        DocumentBuilder dBuilder = factory.newDocumentBuilder();

        Document doc = dBuilder.newDocument();

        //root element

        Element root = document.createElement("root_tag");

        document.appendChild(root);

      //first element

        Element firstElement = document.createElement("element_tag");

        root.appendChild(element);

      //first element first child

        Element firstElementFirstChild =
document.createElement("first_child_of_element_tag");

        firstElement.appendChild(first_child_of_element_tag);

      //first element second child

        Element firstElementSecondChild =
document.createElement("second_child_of_element_tag");

        firstElement.appendChild(second_child_of_element_tag);

//per attaccare nuovi child al root basta usare root.appendChild(Element);
```

```
        TransformerFactory transformerFactory =
TransformerFactory.newInstance();

        Transformer transformer = transformerFactory.newTransformer();

        DOMSource domSource = new DOMSource(document);

        StreamResult streamResult = new StreamResult(new
File("path_output"));

        transformer.transform(domSource, streamResult);



    }catch(Exception ignored){}

}
```

APP LATO CLIENT

Come piattaforma si e' deciso dando per scontate le conoscenze, di utilizzare Android.

In quanto dovremo fare un parsing dei dati xml per, creare una rappresentazione di ogni singolo elemento, andiamo a creare una classe Wrapper che rappresentera' il libro:

```
public class Book implements Serializable {



    private String isbn;

    private String title;

    private String ean;

    private float price;

    private String [] authors;

    private String publisher;
```

```
    private String audienceAge;

    private String releaseDate;

    private String studio;

    private short pages; //32k should be enough ;)

    private short weight;

    //todo change rating type from short to float

    private float rating;

    private String [] genres;

    private String summary;

    private String bookCover;

    private String language;

    private String country;

    private String dewey;

    //non sono presenti i getter ed i setter

}
```

[ATTENZIONE] Si noti l'aggiunta di una stringa (String bookCover) la quale conterra' un immagine in Base64 la quale rappresenta per l'appunto la copertina del libro (tale modifica avra' effetto anche sulla struttura XML la quale conterra' ora un elemento <book_cover> BASE64String </book_cover> all'interno dell'elemento <book> </book>.

Andiamo quindi a vedere la parte di codice che si occupa della connessione al server :

```
public class ClientConnection extends Thread {

    private boolean canRead = false;

    public ClientConnection(){

    }

    @Override
```

```
    public void run() {

        try {

            Socket socket = new Socket("indirizzo ipv4", 7777);

            BufferedReader br = new BufferedReader(new
InputStreamReader(socket.getInputStream()));

            String xmlString = br.readLine();

            socket.close();

            BooksManager bm = BooksManager.getInstance();

            bm.decode(xmlString);

            canRead = true;

        } catch (IOException ignored) {}

    }

    public boolean isReadable(){

        return canRead;

    }

}
```

Si nota l'utilizzo di una variabile booleana che fara' da flag nel metodo che aspetta i dati in modo tale da mettere l'app in uno stato di pausa fin quando tutti i dati non sono stati letti e "parsati".

La classe che si occupera' di gestire i libri e' la classe BooksManager della quale andiamo ad osservare il metodo che effettua il parsing:

```
private final ArrayList <Book> books = new ArrayList<>();

public void decode(String xmlString){

    try {

        books.clear();
```

```
        DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();

        DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

        Document doc = dBuilder.parse(new InputSource(new
StringReader(xmlString)));

doc.getDocumentElement().normalize();

NodeList cList = doc.getElementsByTagName("collection");

for(int x = 0; x < cList.getLength(); x ++) {

Node tList = cList.item(x);

NodeList nList = ((Element) tList).getElementsByTagName("book");

for (int temp = 0; temp < nList.getLength(); temp++) {

Book tBook = new Book();

Node nNode = nList.item(temp);

if (nNode.getNodeType() == Node.ELEMENT_NODE) {

Element e = (Element) nNode;

String title = e.getElementsByTagName("title").item(0).getTextContent();

 if (title.length() > 52) title = title.substring(0,52) + "...";

tBook.setDewhy(e.getAttribute("full_dewhy"));
tBook.setIsbn(e.getElementsByTagName("isbn").item(0).getTextContent());

tBook.setTitle(title);
tBook.setEan(e.getElementsByTagName("ean").item(0).getTextContent());
tBook.setPrice(Float.parseFloat(e.getElementsByTagName("price").item(0).ge
tTextContent().trim()));
tBook.setPublisher(e.getElementsByTagName("publisher").item(0).getTextCont
ent());
tBook.setAudienceAge(e.getElementsByTagName("audience_age").item(0).getTex
tContent());
```

```
tBook.setReleaseDate(e.getElementsByTagName("release_date").item(0).getTex
tContent());
tBook.setStudio(e.getElementsByTagName("studio").item(0).getTextContent())
;
tBook.setPages(Short.parseShort(e.getElementsByTagName("pages").item(0).ge
tTextContent().trim()));
tBook.setWeight(Short.parseShort(e.getElementsByTagName("weight").item(0).
getTextContent().trim()));
tBook.setBookCover(e.getElementsByTagName("book_cover").item(0).getTextCon
tent());
tBook.setRating(Float.parseFloat(e.getElementsByTagName("rating").item(0).
getTextContent().trim()));
tBook.setSummary(e.getElementsByTagName("summary").item(0).getTextContent(
));
tBook.setCountry(e.getElementsByTagName("country").item(0).getTextContent(
));
tBook.setLanguage(e.getElementsByTagName("language").item(0).getTextConten
t());

NodeList authors = ((Element) nNode).getElementsByTagName("author");

NodeList genres = ((Element) nNode).getElementsByTagName("genre");

String[] auths = new String[authors.getLength()];

String[] gens = new String[genres.getLength()];

            for (int i = 0; i < authors.getLength(); i++) {

                auths[i] = authors.item(i).getTextContent();

             }

             for (int i = 0; i < genres.getLength(); i++) {

                   gens[i] = genres.item(i).getTextContent();

             }

             tBook.setAuthors(auths);

             tBook.setGenres(gens);
```

```
                books.add(tBook);

                    }

                }

            }

        } catch (Exception e) {

            e.printStackTrace();

        }

}
```

Il restante del codice lato client e' presente nella cartella XMLEsercizioTurturro\Libraary\app\src\main\java\space\nullpointerexception\libraary.

Mentre, il restante del codice lato server e' presente nella cartella XMLEsercizioTurturro\LibraaryServer\src.

Turturro Vincenzo 1/11/2020