Programare multiparadigmă - **JAVA**

Conf. univ. dr. **Claudiu Vințe**

claudiu.vinte@ie.ase.ro

XML *– Extensible Markup Language*

- XML este un standard pentru explicitarea structurii documentelor folosind un limbaj de descriere
- Caracteristici:
  - Format text pentru descrierea datelor ceea ce-l face simplu de utilizat si ușor editabil
  - Standardizat – există API-uri specializate pentru parsare în majoritatea limbajelor
  - Aplicațiile care utilizează XML trebuie sa stabilească doar partea semantică a reprezentării datelor
- Elementele limbajului:
  - ***Tag***: conțin meta-informații legate de structura si semantica datelor

    <tag>*conținut****</tag>*** *sau* ***<tag />***

  - ***Attribute***: sunt definite prin perechi nume-valoare in interiorul unui tag

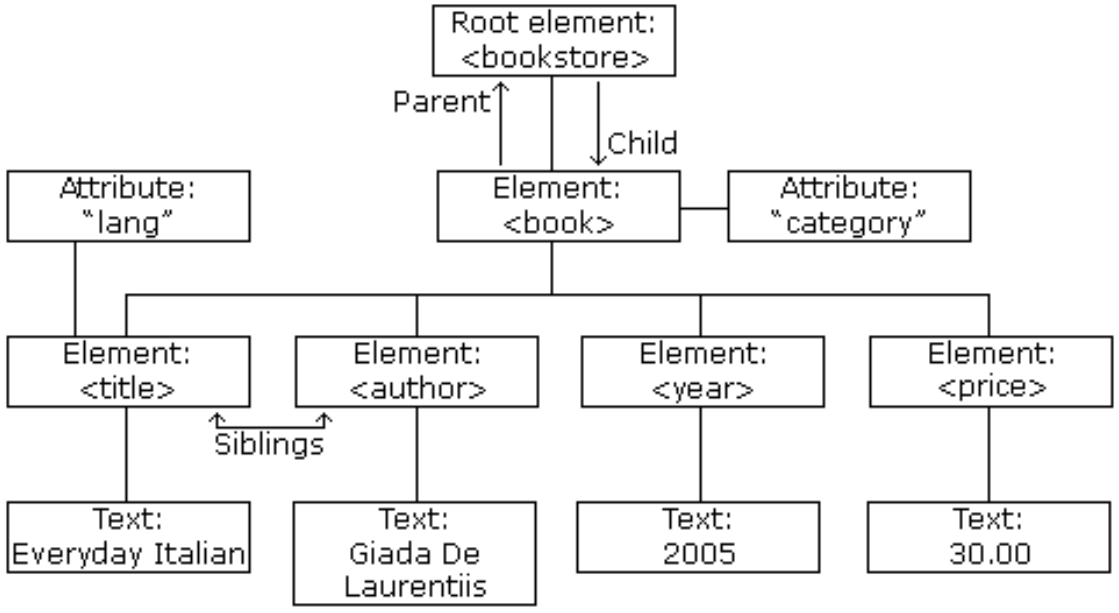
    <tag ***nume="valoare"****>conținut</tag>*

DOM *– Document Object Model*

- In tehnologia DOM, toate elementele unui document XML sunt considerate noduri.
- Tipuri de noduri:
  - Întregul document este un nod-document (clasa Document)
  - Fiecare element XML este un nod-element (clasa Element)
  - Textul XML din elementele XML sunt de tip nod-text (clasa Text)
  - Comentariile sunt de tip nod-comentariu (clasa Comment)
  - Atributele sunt de tip nod-atribut (clasa Attr)
- DOM gestionează un arbore de noduri având ca rădăcina nodul Document.
- Dezavantaj - consum de timp si de memorie
- Avantaj - are o arhitectură arborescentă ușor de înțeles.
- Structura arborescentă operează cu noțiunile de părinte (*parent*), fiu (*children*), frate (*siblings*), astfel:
  - nodul rădăcina este numit *root*
  - fiecare nod cu excepția rădăcinii are un singur părinte
  - un nod poate avea oricâți fii

DOM *– Document Object Model*

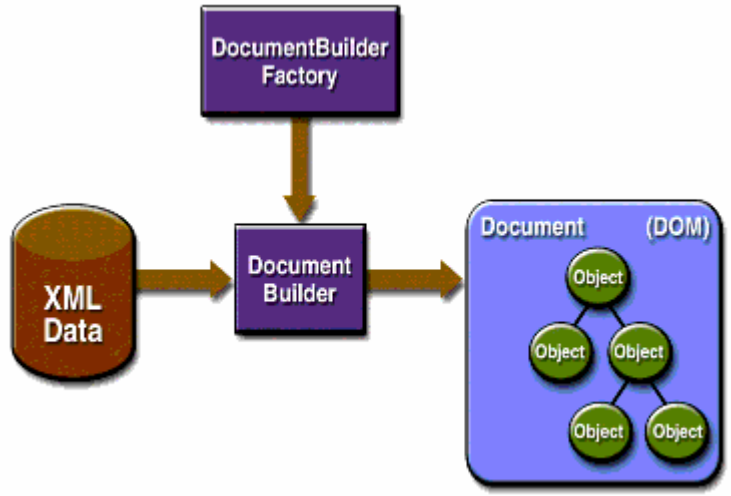
```xml
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
    <book category="cooking">
        <title lang="en">Everyday Italian</title>
        <author>Giada De Laurentiis</author>
        <year>2005</year>
        <price>30.00</price>
    </book>
    <book category="children">
        <title lang="en">Harry Potter</title>
        <author>J K. Rowling</author>
        <year>2005</year>
        <price>29.99</price>
    </book>
    <book category="web">
        <title lang="en">Learning XML</title>
        <author>Erik T. Ray</author>
        <year>2003</year>
        <price>39.95</price>
    </book>
</bookstore>
```



***JAXP**: Java API for XML*

- ***DocumentBuilderFactory*** – clasă de tip *factory* pentru crearea de parsere XML DOM
  - *static DocumentBuilderFactory* ***newInstance****()*: permite instanțierea unui obiect de tip *DocumentBuilderFactory*
  - *abstract DocumentBuilder* ***newDocumentBuilder****():* permite construirea unui obiect de tip *DocumentBuilder*

- ***DocumentBuilder*** – o clasă prin care se pot obține obiectele de tip *Document* asociate unui document XML
  - *Document* ***parse****(File/InputStream/string)*: obține obiectul *Document* prin parsarea fișierului XML specificat
  - *abstract Document* ***newDocument****()*: creare obiect *Document* gol



***JAXP**: Java API for XML*

- ***Node*** – interfața care reprezintă un nod din arbore (indiferent de tip)
  - *Node* ***appendChild****(Node newChild)*: adaugă un nod copil
  - *String* ***getNodeName****()*: furnizează numele nodului
  - *String* ***getNodeValue****()*: întoarce valoarea nodului
  - *short* ***getNodeType****()*: tipul nodului. (constante in interfața *Node*. Exemplu: *Node.ATTRIBUTE_NODE*)
  - *Node* ***getParentNode****()*: nodul părinte
  - *NodeList* ***getChildNodes****()*: lista fiilor
  - *NamedNodeMap* ***getAttributes****()*: lista atributelor
- ***Element*** - interfață derivată din *Node*
  - utilizată pentru descrierea elementelor dintr-un arbore DOM
  - elementele pot avea asociate atribute
  - conține metode de adăugare/consultare/ștergere atribute

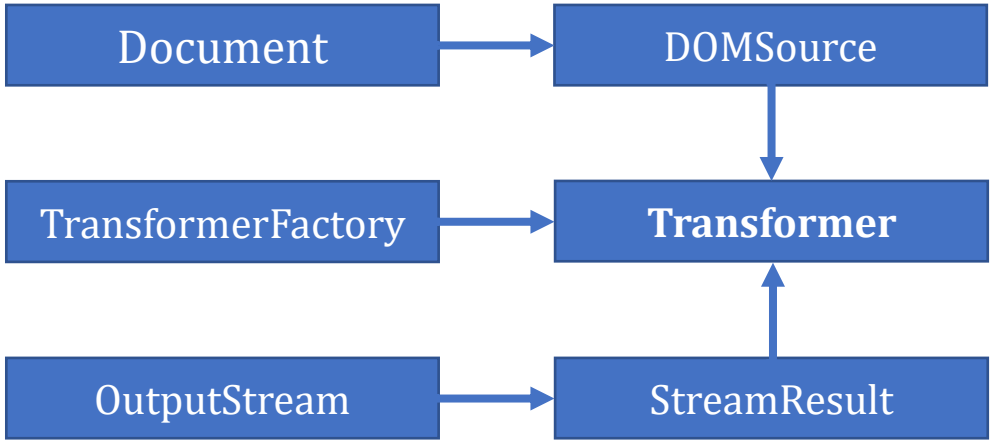
***JAXP**: Java API for XML*

- ***Document*** – interfața care modelează conceptul de document DOM
  - *Element* ***getDocumentElement****()*: întoarce elementul rădăcina
  - *NodeList* ***getElementsByTagName****(String tagname)*: întoarce elementele (din clasa Element) asociate cu numele de tag specificat
  - *Element* ***createElement****(String tagName)*: creează un nod de tip Element cu numele specificat
  - *Attr* ***createAttribute****(String name)*: creează un nod de tip Attr (atribut) cu numele specificat
  - *Comment* ***createComment****(String data)*: creează un nod de tip Comment cu conținutul specificat
  - *Text* ***createTextNode****(String data)*: creează un nod de tip Text cu conținutul specificat
- **Colecții**:
  - *NodeList*: colecție (listă) de noduri cu metode *item* și getL*ength*
  - *NamedNodeMap*: dicționar de noduri cu metode *item,* getL*ength și getNamedItem*

***JAXP**: Java API for XML*

- ***TransformerBuilderFactory*** – clasă de tip *factory* pentru crearea de obiecte de tip *Transformer*
  - *static TransformerBuilderFactory* ***newInstance****()*: permite instanțierea unui obiect de tip *TransformerBuilderFactory*
  - *abstract Transformer* ***newTransformer****():* permite construirea unui obiect de tip *DocumentBuilder*

- ***Transformer*** – o clasă prin care se pot obține obiectele de tip *Document* asociate unui document XML
  - *void* ***setOutputProperty****(String name, String value)*: modificare proprietăți transformare (vezi *OutputKeys*)
  - *void transform(Source xmlSource, Result outputTarget)*: transformare document DOM



DOMSource – obiect sursă de tip DOM pentru o transformare (de tip *Source*)

StreamResult – obiect destinație de tip *stream* pentru o transformare (de tip *Result*)

JSON *- JavaScript Object Notation*

- Este un format pentru interschimb de date bazat pe sintaxa JavaScript
- Un document JSON este compus din elemente de tip:
  - *object* – dicționare de forma *{"cheie1" : valoare1, "cheie2" : valoare2 }*
  - *array* – liste de valori de forma *[valoare1, valoare2]*
  - Valori simple de tip: *string, number, true, false* și *null* sau compuse (*object*, *array*)
- Documentație și biblioteci pentru diverse limbaje: https://www.json.org/
- Exemplu de bibliotecă JSON pentru Java: https://github.com/stleary/JSON-java

Biblioteca JSON-Java

- ***JSONObject*** **–** colecție **neordonată** de perechi cheie-valoare; cheile sunt de tip *String* iar valorile pot fi oricare dintre următoarele tipuri: *Boolean*, *JSONArray*, *JSONObject*, *Number*, *String:*
  - *JSONObject()* - creează un obiect gol;
  - *JSONObject(JSONTokener x)* - creează un obiect pornind de la un *JSONTokener*; utilizat de obicei pentru preluarea informațiilor din fișiere;
  - *JSONObject(java.util.Map<?,?> map)* - creează un obiect *JSONObject* pornind de la un obiect *Map* pe baza cheilor și valorilor din Map;
  - Metode de tip ***get*** și ***opt*** pentru citirea valorilor (exemple: *getDouble(String), optString(String),* **...**);
  - Metoda de tip ***put*** pentru adăugarea / modificarea valorilor;
  - *java.util.Set<java.lang.String> keySet()*: întoarce cheile într-un obiect de tip Set;
  - *int length()*: furnizează numărul de chei;
  - *java.io.Writer write(java.io.Writer writer)* și *java.io.Writer write(java.io.Writer writer, int indentFactor, int indent)*: metodele write scriu obiectul în format JSON într-un flux Writer (opțional cu indentare)

Biblioteca JSON-Java

- ***JSONArray*** - secvență **ordonată** de valori (*Boolean, JSONArray, JSONObject, Number, String*)
  - *JSONArray()* - creează un obiect gol;
  - *JSONArray(JSONTokener x)* - creează un obiect pornind de la un *JSONTokener*; utilizat de obicei pentru preluarea informațiilor din fișiere;
  - *JSONArray(java.util.Collection<?,?> col)* - creează un obiect *JSONArray* pe baza valorilor dintr-un obiect de tip *Collection*;
  - Metode de tip ***get*** și ***opt*** pentru citirea valorilor (exemple: *getDouble(int), optString(int),* ...);
  - Metoda de tip ***put*** pentru adăugarea / modificarea valorilor;
  - *int length()*: furnizează numărul de chei;
  - *java.io.Writer write(java.io.Writer writer)* și *java.io.Writer write(java.io.Writer writer, int indentFactor, int indent)*: metodele write scriu obiectul în format JSON într-un flux Writer (opțional cu indentare)

Biblioteca JSON-Java

- ***JSONTokener*** **–** obiect care permite conversia unui text în secvență de elemente JSON; utilizat pentru construirea obiectelor de tip *JSONArray* și *JSONObject*:
  - JSONTokener(Reader reader): construiește un obiect pe baza unui flux de caractere;
  - JSONTokener(InputStream stream): construiește un obiect pe baza unui flux binar;
  - JSONTokener(String json): construiește un obiect pe baza unui text JSON;

- Conversie obiect *JSONArray* în *stream* de obiecte:

```
JSONArray array = new JSONArray();
// ...
StreamSupport
        .stream(array.spliterator(), false)
        .map(item -> (JSONObject)item)
        ...
```