

SER : LABORATOIRE #3

Johanna Melly & Yohann Meyer

Professeur
Eric Lefrançois

18 mai 2018

Table des matières

1	Introduction	2
2	Modification éventuelles	2
3	Objectifs	2
3.1	JSON Schema	2
3.2	XML grammar	3
3.3	printscreen site	7
4	Conclusion	9

1 Introduction

Deux objectifs dans ce labo. Le premier, écrire la grammaire JSON Schema permettant de valider la grammaire de notre JSON créé dans le labo 2. Le second, transformer la structure du fichier XML précédemment (et savamment) construit en un XML contenant les mêmes informations, mais respectant une structure XML prédéfinie par le professeur.

2 Modification éventuelles

Aucune. Little bit disappointed.

3 Objectifs

3.1 JSON Schema

Après avoir savamment étudié la structure de notre schema JSON précédemment construit, nous avons construit le fichier de grammaire adapté.
Pas de commentaires à faire là dessus.

```
{
  "$schema" : "http://json-schema.org/schema#",
  "type" : "array",
  "properties" :
  {
    "id" :
    {
      "type" : "integer"
    },
    "dateHeure" :
    {
      "type" : "object",
      "properties" :
      {
        "year" : { "type": "integer" },
        "month" : { "type": "integer" },
        "dayOfMonth" : { "type": "integer" },
        "hourOfDay" : { "type": "integer" },
        "minute" : { "type": "integer" },
        "second" : { "type": "integer" }
      }
    },
    "film":
    {
      "type": "object",
      "properties":
      {
        "Acteurs":
        {
          "type": "object",
          "properties":
          {
            "Role1": { "type": "string" },
            "Role2": { "type": "string" }
          }
        },
        "Titre": { "type": "string" }
      }
    }
  }
}
```

```

public final class Essai {
    public static void main(final String... args)
        throws IOException, ProcessingException {
        JsonNode schema_structure = JsonLoader.fromFile(new File( s: "jsonschema.schema.json"));
        JsonNode structure_a_valider= JsonLoader.fromFile(new File( s: "Projections.json"));

        JsonSchemaFactory factory = JsonSchemaFactory.byDefault();
        JsonSchema schema = factory.getJsonSchema(schema_structure);

        ProcessingReport report = schema.validate(structure_a_valider);

        Essai > main()
}

```

Essai

```

/usr/lib/jvm/java-8-openjdk-amd64/bin/java ...
*** La structure est valide! ***

com.github.fge.jsonschema.core.report.ListProcessingReport: success

Process finished with exit code 0

```

FIGURE 1 – validation JSON

3.2 XML grammar

En passant au-dessus des cotés fastidieux et plutôt long du travail demandé, le principal point de tension fut la récupération des ID de nos champs, n'ayant pas utilisé d'IDREF dans notre conception lors du labo précédent.

Du coup, la liste d'acteur récupère toutes les informations propre à un acteur directement dans la structure de projections à la place d'avoir une liste extérieure qui serait plus propre, une amélioration pour la prochaine itération.

On trouvera le résultat de la validation du XML dans un (long) fichier Output_validation.txt Code XML

```

import com.sun.org.apache.bcel.internal.generic.GETFIELD;
import org.jdom2.*;
import org.jdom2.filter.Filters;
import org.jdom2.output.Format;
import org.jdom2.output.XMLOutputter;
import org.jdom2.xpath.XPathExpression;
import org.jdom2.xpath.XPathFactory;

import java.io.FileWriter;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

public class main {
    public static void main(String[] args) {
        String xmlFile = "proj.XML";
        Document document = XMLGen.getDOMParsedDocument(xmlFile);

        Element rootNode = document.getRootElement();
        System.out.println("Root Element: " + rootNode.getName());
        XPathFactory xpfac = XPathFactory.instance();
        XPathExpression<Element> projections = xpfac.compile("//Projection",
            Filters.element());
        Element plex = new Element("plex");
        // Creation du document XML avec Projections comme racine
        DocType dtype = new DocType(plex.getName());
        dtype.setSystemID("projections.dtd");

        Document plex_doc = new Document(plex, dtype);

        // On defini la feuille xslt
        Map<String, String> map = new HashMap<String, String>();
        map.put("href", "projections.xsl");
        map.put("type", "text/xsl");
        plex_doc.addContent(0, new ProcessingInstruction("xml-stylesheet", map));
    }
}

```

```

Element projectionsElement = new Element("projections");
Element filmsElement = new Element("films");
Element acteursElement = new Element("acteurs");
Element listeLangagesElement = new Element("liste_langages");
Element listeGenresElement = new Element("liste_genres");
Element listeMotsClesElement = new Element("liste_mots_cles");

for (Element e : projections.evaluate(document))
{
    // =====
    // ===== PROJECTIONS =====
    // =====

    // ===== PROJECTION =====
    Element projectionElement = new Element("projection");
    projectionElement.setAttribute("film_id", "F" +
        e.getChild("Film").getAttribute("Id").getValue());
    projectionElement.setAttribute("titre",
        e.getChild("Film").getChildText("titre"));

    // ===== SALLE =====
    Element salleElement = new Element("salle");
    salleElement.setAttribute("taille",
        e.getChild("Salle").getChildText("taille"));
    salleElement.setText(e.getChild("Salle").getChildText("numero"));
    projectionElement.addContent(salleElement);

    // ===== DATEHEURE =====
    Element dateHeureElement = new Element("date_heure");
    dateHeureElement.setAttribute("format", "dd.MM.YYYY HH:mm");
    String formatDate = e.getChild("Date").getChildText("jour") + ".";
    formatDate += e.getChild("Date").getChildText("mois") + ".";
    formatDate += e.getChild("Date").getChildText("annee").substring(2);
    formatDate += " 00:00";
    dateHeureElement.setText(formatDate);
    projectionElement.addContent(dateHeureElement);

    projectionsElement.addContent(projectionElement);

    // ===== FILM =====
    Element filmElement = new Element("film");
    filmElement.setAttribute("no", "F" +
        e.getChild("Film").getAttribute("Id").getValue());
    // => TITRE
    Element titreElement = new Element("titre");
    titreElement.setText(e.getChild("Film").getChildText("titre"));
    filmElement.addContent(titreElement);
    // => DUREE
    Element dureeElement = new Element("duree");
    dureeElement.setAttribute("format", "minutes");
    dureeElement.setText(e.getChild("Film").getChildText("duree"));
    filmElement.addContent(dureeElement);
    // => SYNOPSIS
    Element synopsisElement = new Element("synopsis");
    synopsisElement.setText("<![CDATA[" +
        e.getChild("Film").getChildText("synopsis") + "]]>");
    filmElement.addContent(synopsisElement);
    // => PHOTO
    if(e.getChild("Film").getChild("photo") != null){
        Element photoElement = new Element("photo");
        photoElement.setAttribute("url",
            e.getChild("Film").getChild("photo").getAttributeValue("url"));
        filmElement.addContent(photoElement);
    }
}

```

```

// => CRITIQUES
Element critiquesElement = new Element("critiques");
for(Element cri : e.getChild("Film").getChildren("Critique")){
    Element critiqueElement = new Element("critique");
    critiqueElement.setAttribute("note", cri.getAttributeValue("Note"));
    critiqueElement.setText(cri.getText());
    critiquesElement.addContent(critiqueElement);
}
if(critiquesElement.getChildren() != null){
    filmElement.addContent(critiquesElement);
}

// => LANGAGES
Element langagesElement = new Element("langages");
String listeLangages = "";
for(Element lan : e.getChild("Film").getChildren("Langage")){
    listeLangages += "L" + lan.getAttributeValue("Id") + " ";
}
langagesElement.setAttribute("liste", listeLangages);
filmElement.addContent(langagesElement);

// => GENRES
Element genresElement = new Element("genres");
String listeGenres = "";
for(Element gen : e.getChild("Film").getChildren("Genre")){
    listeGenres += "G" + gen.getAttributeValue("Id") + " ";
}
genresElement.setAttribute("liste", listeGenres);
filmElement.addContent(genresElement);

// => MOTS CLES
Element motsclesElement = new Element("mots_cles");
String listeMotsCles = "";
for(Element mot : e.getChild("Film").getChildren("Motcle")){
    listeMotsCles += "M" + mot.getAttributeValue("Id") + " ";
}
motsclesElement.setAttribute("liste", listeMotsCles);
filmElement.addContent(motsclesElement);

// => ROLES
Element rolesElement = new Element("roles");
for(Element rol : e.getChild("Film").getChildren("Role")){
    Element roleElement = new Element("role");
    roleElement.setAttribute("place", rol.getChildText("place"));
    roleElement.setAttribute("personnage", rol.getChildText("personnage"));
    roleElement.setAttribute("acteur_id", "A"+
        rol.getChild("Acteur").getAttributeValue
            ("Id").substring(6,
                rol.getChild("Acteur").getAttributeValue("Id").length()));
    rolesElement.addContent(roleElement);
}
filmElement.addContent(rolesElement);

filmsElement.addContent(filmElement);

// ===== ACTEURS
=====
for (Element rol : e.getChild("Film").getChildren("Role")){
    Element act = rol.getChild("Acteur");
    Element acteurElement = new Element("acteur");
    acteurElement.setAttribute("no", "A"+
        act.getAttributeValue("Id").substring
            (6, act.getAttributeValue("Id").length()));
    // => NOM
    Element nomActeurElement = new Element("nom");
    nomActeurElement.setText(act.getChildText("nom"));
    acteurElement.addContent(nomActeurElement);
}

```

```

// => NOM NAISSANCE
Element nomNaissanceActeurElement = new Element("nom_naissance");
if(act.getChild("nomNaissance") != null){
    nomNaissanceActeurElement.setText(act.getChildText("nomNaissance"));
}else{
    nomNaissanceActeurElement.setText("n/a");
}
acteurElement.addContent(nomNaissanceActeurElement);
// => SEXE
Element sexeActeurElement = new Element("sexe");
sexeActeurElement.setAttribute("valeur", act.getAttributeValue("sexe"));
acteurElement.addContent(sexeActeurElement);

// => DATE NAISSANCE
Element dateNaissanceActeur = new Element("date_naissance");
dateNaissanceActeur.setAttribute("format", "dd.mm.yyyy");
if(act.getChild("dateNaissance") != null) {

    String[] date = act.getChildText("dateNaissance").split("/");
    dateNaissanceActeur.setText(date[0] + "." + date[1] + "." + date[2]);
}
acteurElement.addContent(dateNaissanceActeur);

// => DATE DECES
Element dateDecesActeur = new Element("date_decès");
dateDecesActeur.setAttribute("format", "dd.mm.yyyy");
if(act.getChild("dateDeces") != null){

    String[] date2 = act.getChildText("dateDeces").split("/");
    dateDecesActeur.setText(date2[0] + "." + date2[1] + "." + date2[2]);
}
acteurElement.addContent(dateDecesActeur);

// => BIOGRAPHIE
Element biographieActeurElement = new Element("biographie");
if(act.getChild("biographie") != null){
    biographieActeurElement.setText("<![CDATA[" +
        act.getChildText("biographie") + "]]>");
}
acteurElement.addContent(biographieActeurElement);

acteursElement.addContent(acteurElement);
}

// ===== LISTE LANGAGES
=====

for(Element lan : e.getChild("Film").getChildren("Langage")){
    Element langageElement = new Element("langage");
    langageElement.setAttribute("no", "L" + lan.getAttributeValue("Id"));
    langageElement.setText(lan.getChildText("Label"));
    listeLangagesElement.addContent(langageElement);
}

// ===== LISTE GENRES =====
for(Element gen : e.getChild("Film").getChildren("Genre")){
    Element genreElement = new Element("genre");
    genreElement.setAttribute("no", "G" + gen.getAttributeValue("Id"));
    genreElement.setText(gen.getChildText("Label"));
    listeGenresElement.addContent(genreElement);
}

// ===== LISTE MOTS CLES =====
for(Element mot : e.getChild("Film").getChildren("Motcle")){
    Element motCleElement = new Element("mot_cle");
    motCleElement.setAttribute("no", "M" + mot.getAttributeValue("Id"));

```

```

        motCleElement.setText(mot.getChildText("label"));
        listeMotsClesElement.addContent(motCleElement);
        System.out.println(motCleElement.getText());
        System.out.println(motCleElement.getAttributeValue("no"));
    }

}

plex.addContent(projectionsElement);
plex.addContent(filmsElement);
plex.addContent(acteursElement);
plex.addContent(listeLangagesElement);
plex.addContent(listeGenresElement);
plex.addContent(listeMotsClesElement);

XMLOutputter xmlOutput = new XMLOutputter();

xmlOutput.setFormat(Format.getPrettyFormat());
// Ecriture dans un fichier
try{
    xmlOutput.output(plex_doc, new FileWriter("./projections.xml"));
}catch(IOException e){
    System.out.println(e.getMessage());
}

//Read employee first names
/*
XPathExpression<Element> xPathN =
    xpfac.compile("//employees/employee/firstName", Filters.element());

for (Element element : xPathN.evaluate(document))
{
    System.out.println("Employee First Name :: " + element.getValue());
}
*/
}
}

```

3.3 printscreen site

Plex - Projections			
Film	Projection	Salle	Note
Casino Royale	16.3.18 00:00	Flon 1	NaN/5.0

FIGURE 2 – page accueil

Plex - Projections

Casino Royale



Durée : 144 minutes

Genres : Action - Thriller - Adventure

Langues : French - English

Synopsys

<![CDATA[Casino Royale introduces James Bond before he holds his license to kill. But Bond is no less dangerous, and with two professional assassinations in quick succession, he is elevated to '00' status. Bond's first 007 mission takes him to Uganda where he is to spy on a terrorist, Mollaka. Not everything goes to plan and Bond decides to investigate, independently of MI6, in order to track down the rest of the terrorist cell. Following a lead to the Bahamas, he encounters Dimitrios and his girlfriend, Solange. He learns that Dimitrios is involved with Le Chiffre, banker to the world's terrorist organizations. Secret Service intelligence reveals that Le Chiffre is planning to raise money in a high-stakes poker game in Montenegro at Le Casino Royale. MI6 assigns 007 to play against him, knowing that if Le Chiffre loses, it will destroy his organization. 'M' places Bond under the watchful eye of the beguiling Vesper Lynd. At first skeptical of what value Vesper can provide, Bond's interest in her deepens as they brave danger together and even torture at the hands of Le Chiffre. In Montenegro, Bond allies himself with Mathis MI6's local field agent, and Felix Leiter who is representing the interests of the CIA. The marathon game proceeds with dirty tricks and violence, raising the stakes beyond blood money and reaching a terrifying climax.]]>

Critiques

Rôles

Place	Personnage	Acteur
1	James Bond	Craig, Daniel
2	Vesper Lynd	Green, Eva
3	Le Chiffre	Mikkelsen, Mads
4	M	Dench, Judi
5	Felix Leiter	Wright, Jeffrey
6	Rene Mathis	Giannini, Giancarlo
7	Solange	Murino, Caterina
8	Alex Dimitrios	Abkarian, Simon
9	Steven Obanno	De Bankolé, Isaach
10	Mr. White	Christensen, Jesper
11	Valenka	Milicevic, Ivana

FIGURE 3 – détails du film 1

67	Venice Hotel Concierge	G'Vera, Ivan
68	Hotel Splendide Limo Driver	Lenc, Jiri
69	Hermitage Waiter	Jankovsky, Jaroslav
9223372036854775807	Airport Staff	Atkins, Lasco

Mots-cles

returning-character-with-different-actor	british-secret-service	slow-motion-scene	prequel-to-sequel	security-camera	off-screen-murder	horse-and-carriage	secret-device							
action-hero	penumbra	horse	martial-arts	army	martini	cocktail	killed-in-an-elevator	jungle	aston-martin	poisoned	miami-florida	exploding-bus	corpse	prisoner
secret-service	based-on-novel	sequel-mentioned-during-end-credits				gambling-syndicate	london-england	machine-gun	stairwell	car-accident	car-chase	dead-woman		
body-bag	cell-phone	playing-cards	darkroom	chase	wager	foot-pursuit	villa	lifting-someone-into-the-air	damsel-in-distress	banker	airport	hit-in-the-crotch		
corporal-punishment	bomb	shower	motorboat	strangulation	train	foot-chase	drowning	pistol	animated-credits	infidelity	super-villain	terrorist-base	spa	
sailing-boat	beach	pinball-machine	money	title-spoken-by-character	henchman	psychopath	bond-girl	shot-in-the-foot	newspaper-headline	crane	shot-in-the-leg			
altered-version-of-studio-logo	tied-to-a-chair	cia-agent	hostage	mass-murder	blockbuster	trapped-in-an-elevator	money-laundering	snake	lagoon	parkour	montenegro			
heart-attack	arch-villain	drugged-drink	lifting-an-adult-into-the-air	origin-story	flashback	cult-film	electrocution	hotel	beating	suspense	violence	poison	airplane	
male-nudity	torture	police-car	beach-resort	neck-breaking	double-cross	hotel-receptionist	femme-fatale	yacht	walther-p99	tanker	swimming	silencer	2000s	
sequel	italy	stabbed-in-the-chest	secret-agent	plot-twist	fire-sprinkler	siren	shot-in-the-eye	lost-love	dead-woman-with-eyes-open	dead-girl	south-africa	free-running		
taser	ballsong	flood	shot-in-the-head	adultery	poker-the-card-game	shotgun	nassau-bahamas	shot-in-the-forehead	gadget-car	explosion	bodyworks	rogue-agent		
murder-by-gunshot	gunfight	blood	ak-47	cult-figure	hospital	casino	computer	uganda	shootout	reboot-of-series	assassination	double-agent	building-collapse	
construction-site	gadget	surveillance	official-james-bond-series	woman-drowned		exploding-body	blood-spatter	eye-patch	terrorist	death	machete	crushed-to-death		
surprise-ending	prague-czech-republic	police-officer	shot-in-the-chest	mongoose	stabbing	fatal-attraction	village	venice-italy	video-camera	black-and-white-prologue				
terrorism	police-arrest	fight	blood-on-shirt	exhibit	soldier	held-at-gunpoint	attempted-poisoning	bahamas	flipping-car	restaurant	shot-in-the-back	africa	tough-guy	
madagascar	hammock	bare-chested-male-bondage	assassin	horse-riding	gambling	embassy	lifting-male-in-air	helicopter	espionage	passport	murder			
laptop-computer														

FIGURE 4 – détails du film 2

Plex - Projections

Green, Eva

Sexe : FEMININ

Nom de naissance : Green, Eva Gaëlle

Date de naissance : 5.6.1980

Date de décès :

Biographie

<![CDATA[Eva Gaëlle Green was born on July 5, 1980, in Paris, France. She has a non-identical twin sister. Her father, Walter Green, is a dentist who appeared in the 1966 film *„Au hasard Balthazar“* (qv). Her mother, „Marlène Jobert“ (qv), is an actress who retired from acting and became a writer of children's books. Eva's mother was born in Algeria, of Sephardi Jewish heritage (during that time, Algeria was part of France), and Eva's father is of Swedish and French descent. Eva left French school at 17. She switched to English in Ramsgate, Kent, and went to the American School in France for one year. She studied acting at Saint Paul Drama School in Paris for three years, then had a 10-week polishing course at the Weber Douglas Academy of dramatic Art in London. She also studied directing at the Tisch School of Arts at New York University. She returned to Paris as an accomplished young actress, and played on stage in several theater productions: *„La Jalousie en Trois Fax“* and *„Turcaret“*. There she caught the eye of director „Bernardo Bertolucci“ (qv). Green followed a recommendation to work on her English. She studied for two months with an English coach before doing *„The Dreamers“* (2003) (qv) with „Bernardo Bertolucci“ (qv). During their work Bertolucci described Green as being *“so beautiful it's indecent.”* Green won critical acclaim for her role in *„The Dreamers“*

FIGURE 5 – détails acteurs

4 Conclusion

Dans ce laboratoire, nous avons perfectionné notre compréhension du XML et du JSON. Il n'était comparativement pas trop compliqué, la structure persistante du labo précédent permettant une plus grande facilité dans la compréhension intuitive de la structure de départ. La jonction d'avec la structure XML demandée ne fut donc pas un problème, sinon de part le temps nécessaire à son implémentation. La partie JSON nous a démontré une fois encore la différence de rapidité d'utilisation/implémentation (bien que les deux objectifs ne soyant pas parfaitement identiques), et de comprendre l'importance d'une bonne *blanquette* sérialisation.