Guidoux Vincent & Jaquet David

Schéma JSON

SER – Laboratoire 3

Table des matières

[1 Petite introduction 2](#_Toc514611897)

[2 Modification effectuée sur la structure XML 2](#_Toc514611898)

[3 Objectif 1 2](#_Toc514611899)

[3.1 Grammaire JSON Schema 2](#_Toc514611900)

[3.2 Résultat de la validation 3](#_Toc514611901)

[4 Objectif 2 3](#_Toc514611902)

[4.1 Principe conceptuel de la solution 3](#_Toc514611903)

[4.2 Code 4](#_Toc514611904)

[4.3 Résultat obtenu sur navigateur web 5](#_Toc514611905)

[5 Petite conclusion 7](#_Toc514611906)

[6 Annexe 7](#_Toc514611907)

[6.1 Fichiers 7](#_Toc514611908)

# Petite introduction

Pour ce laboratoire, nous avions deux objectifs distincts ; le premier est de créer une grammaire JSON Schema permettant de valider le fichier JSON élaboré lors du précédent laboratoire.

Le second objectif est de créer un programme Java s’appuyant sur le fichier XML généré lors du précédent laboratoire. Le programme doit créer un nouveau fichier XML. Le nouveau fichier doit avoir un en-tête précis et doit pouvoir être validé par la DTD fournie.

# Modification effectuée sur la structure XML

Nous nous sommes basés sur fichier XML fourni pour ce laboratoire afin de connaître les éléments qu’il nous manquait dans les fichiers que nous avons généré lors des précédents laboratoires. Nous avons donc rajouté la taille de la salle en attribut. L’objet film a également été modifié afin d’y ajouter l’attribut film\_id. Les genres, les mots-clés, les acteurs ainsi que les langues ont un attribut no qui a également été rajouté. Il s’agit des seules modifications qui ont été apportées par rapport à l’XML du précédent laboratoire.

# Objectif 1

{  
 **"type"**: **"object"**,  
 **"properties"**: {  
 **"projections"**: {  
 **"type"**: **"array"**,  
 **"items"**: {  
 **"type"**: **"object"**,  
 **"properties"**: {  
 **"date"**: {  
 **"type"**: **"object"**,  
 **"properties"**: {  
 **"jour"**: {**"type"**: **"integer"**},  
 **"mois"**: {**"type"**: **"integer"**},  
 **"annee"**: {**"type"**: **"integer"**},  
 **"heure"**: {**"type"**: **"integer"**},  
 **"minute"**: {**"type"**: **"integer"**}  
 }  
 },  
 **"film"**: {  
 **"type"**: **"object"**,  
 **"properties"**: {  
 **"titre"**: {**"type"**: **"string"**},  
 **"acteurs"**: {  
 **"type"**: **"array"**,  
 **"items"**: {  
 **"type"**: **"object"**,  
 **"properties"**: {  
 **"nom\_naissance"**: {**"type"**: **"string"}**,  
 **"role"**: {**"type"**: **"string"**}  
 }  
 }  
 }  
 }  
 }  
 }  
 }  
 }  
 }  
}

*monschema.json*

## Grammaire JSON Schema

Nous avons utilisé le JSON que nous avions générée lors du précédent laboratoire. Le JSON étant déjà correcte, nous n’avons pas effectué de changement sur ce dernier.

On peut voir dans la grammaire ci-contre que nous utilisons un tableau de projections. Chaque projection contient une date et un film.

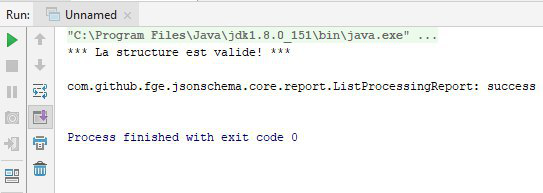
La date est décomposée en sous élément représentant le jour, le mois, l’année, l’heure ainsi que les minutes de la projection. Chacun de ces éléments sont des entiers.

Le film est également séparé en sous éléments. Nous distinguons le titre, qui est du type string, ainsi qu’un tableau d’acteurs.

Les acteurs ne contiennent que leur nom de naissance. Ils sont toutefois des objets car les acteurs de la version JSON sont une version raccourcie par rapport aux acteurs de la version XML conformément à la donnée du laboratoire.

## Résultat de la validation

Comme nous pouvons le voir sur l’image ci-dessous, notre grammaire ne nous génère pas d’erreur et est, par conséquent, valide. L’image est le résultat de l’exécution du programme JSON\_Validation fourni pour ce laboratoire.



# Objectif 2

## Principe conceptuel de la solution

Toute la solution réside dans un seul fichier java Amorce.java qui lit le fichier plex.xml (généré au laboratoire précédent), le parcourt en utilisant Jdom2 et jaxen, prends les informations nécessaires, et recréer un fichier qui correspond à projections.dtd Nous avons mis en évidence un exemple d’utilisation XPath pour aller chercher les informations sur les langues, le reste est exploré en Jdom2. Le fichier .xsl s’attend à un format « dd.MM.YYYY HH:mm » pour les heures, le notre étant moins épuré (une balise pour le jour, une pour le mois.. etc.) nous avons dû faire une petite adaptation dans la lecture du fichier. Nous n’avions pas fait une liste des acteurs dans notre fichier .xml du coup, il y a peut-être des doublons, pour éviter ça dans le fichier nécessaire à ce labo, nous avons fait une ArrayList d’acteurs déjà utilisés en sauvegardant leur id. Comme la note d’un film est une moyenne, cette dernière est considérée comme un nombre réel.

## Code

Par soucis de place, nous avons décidé de n’afficher qu’une partie du code de la fonction createXML du fichier Amorce.java. Nous avons choisi de mettre dans ce rapport la partie contenant les projections ainsi que la partie contenant les langues afin d’avoir un exemple de Jdom2 et de XPath.

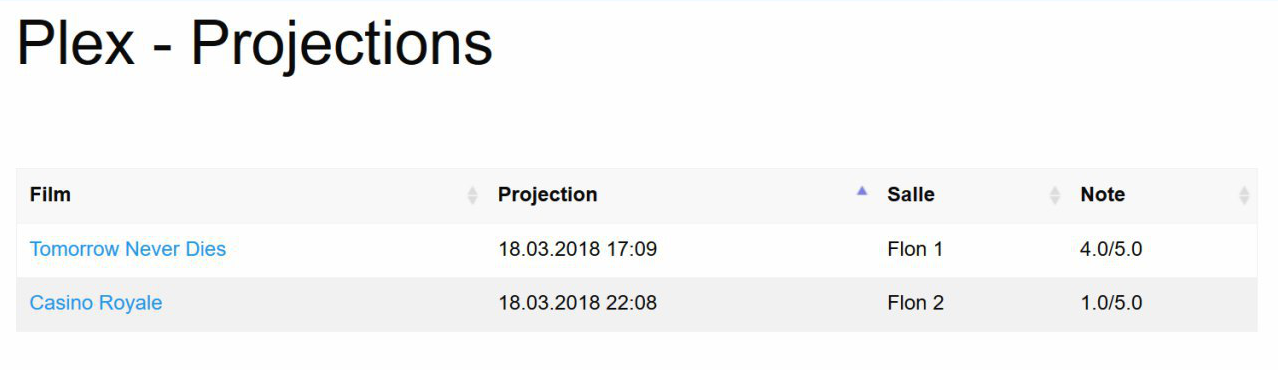
**private static void** createXML(Document labo2) {  
  
 **try** {  
  
 Document document = **new** Document();  
 XMLOutputter outp = **new** XMLOutputter(Format.*getPrettyFormat*());  
  
 *// -- Ajout d'une référence à la DTD* DocType docType = **new** DocType(**"plex"**, **"projections.dtd"**);  
 document.addContent(docType);  
  
 *// -- Ajout de la référence à la feuille XSL* ProcessingInstruction piXSL = **new** ProcessingInstruction(**"xml-stylesheet"**);  
 HashMap<String, String> piAttributes = **new** HashMap<String, String>();  
 piAttributes.put(**"type"**, **"text/xsl"**);  
 piAttributes.put(**"href"**, **"projections.xsl"**);  
 piXSL.setData(piAttributes);  
 document.addContent(piXSL);  
  
 *//Element racine* Element plex = **new** Element(**"plex"**);  
  
 *//---------------------------------------------------------------------------------------------  
 // PROJECTIONS  
 //---------------------------------------------------------------------------------------------* Element projections = **new** Element(**"projections"**);  
  
 **for** (Element projectionTMP : labo2.getRootElement().getChildren(**"projection"**)) {  
 Element projection = **new** Element(**"projection"**);  
  
 Element salle = **new** Element(**"salle"**);  
 salle.setAttribute(**"taille"**, projectionTMP.getChild(**"salle"**).getAttribute(**"taille"**).getValue());  
 salle.addContent(projectionTMP.getChild(**"salle"**).getContent(0).getValue());  
  
 projection.addContent(salle);  
  
 Element date\_heure = **new** Element(**"date\_heure"**);  
 Element date = projectionTMP.getChild(**"date"**);  
  
 date\_heure.setAttribute(**"format"**, **"dd.MM.YYYY HH:mm"**);  
 date\_heure.addContent(*getDateHour*(date));  
  
 projection.addContent(date\_heure);  
  
 projection.setAttribute(**"film\_id"**, projectionTMP.getChild(**"film"**).getAttributeValue(**"film\_id"**));  
 projection.setAttribute(**"titre"**, projectionTMP.getChild(**"film"**).getChild(**"titre"**).getValue());  
  
  
 projections.addContent(projection);  
 }  
  
 plex.addContent(projections);

1

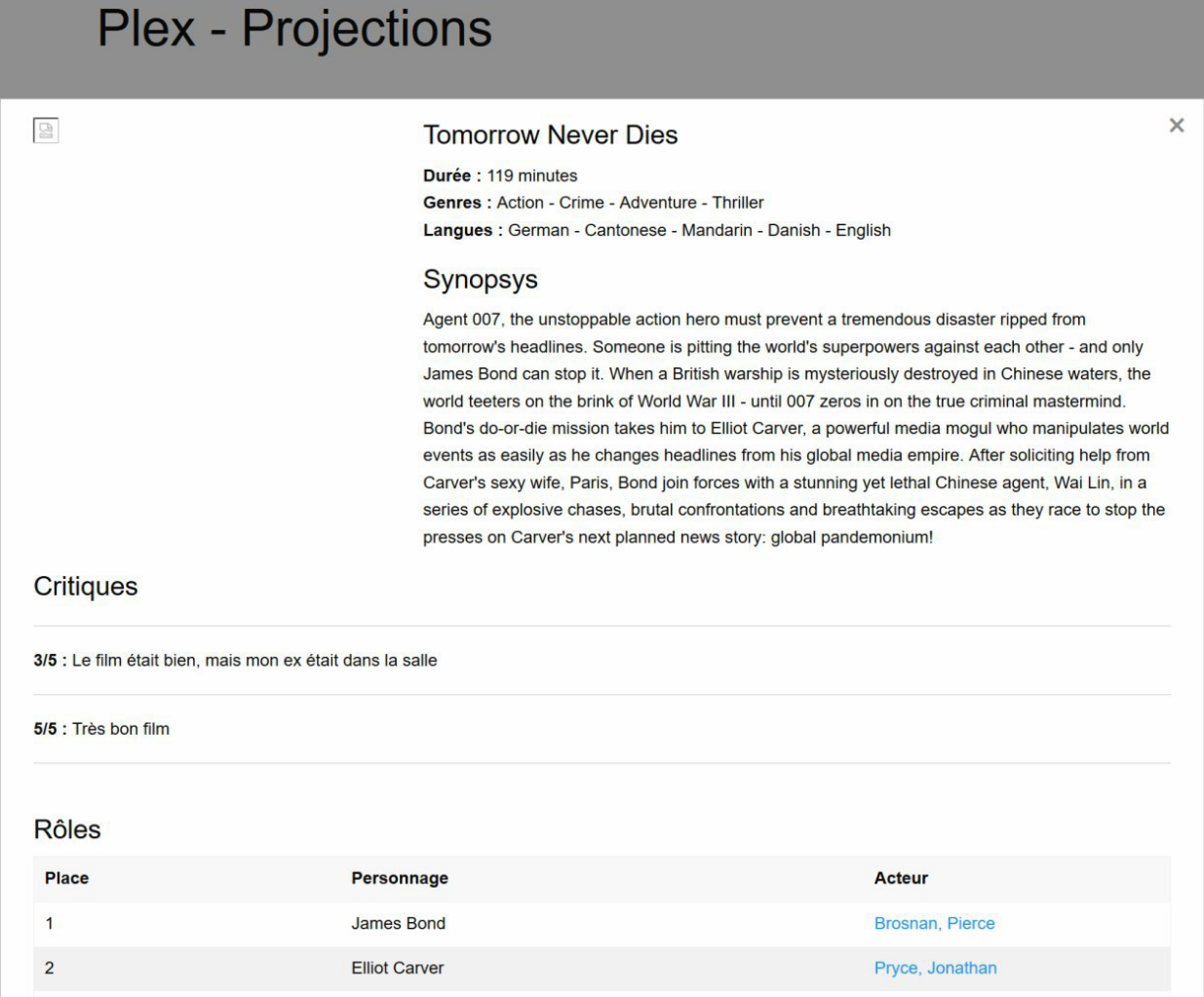
## Résultat obtenu sur navigateur web

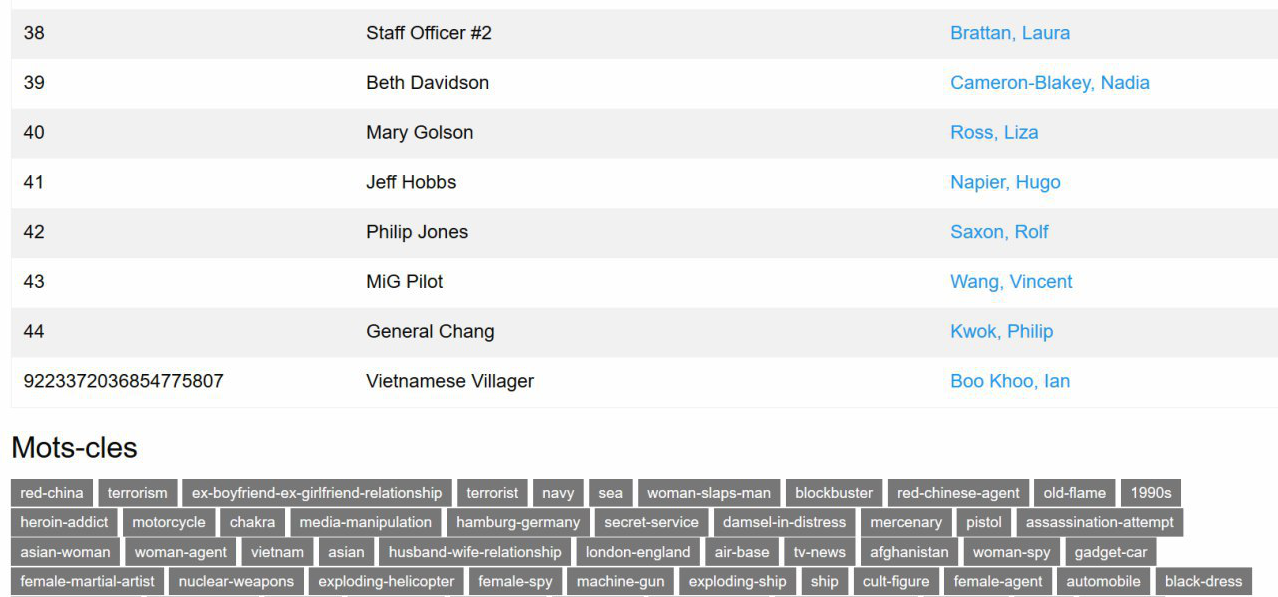
*//----------------------------------------------------------------------------------------------------  
// LISTE\_LANGAGES  
//----------------------------------------------------------------------------------------------------*Element liste\_langages = **new** Element(**"liste\_langages"**);  
  
SAXBuilder builder = **new** SAXBuilder();  
Document xmlDoc = builder.build(**new** File(**"plex.xml"**));  
  
XPathFactory xpfac = XPathFactory.*instance*();  
XPathExpression xp = xpfac.compile(**"//film/langages/langage"**, Filters.*element*());  
  
List<Element> langages = (List<Element>) (xp.evaluate(xmlDoc));  
**for** (Element l : langages) {  
 Element langage = **new** Element(**"langage"**);  
 langage.addContent(l.getValue());  
 langage.setAttribute(**"no"**, l.getAttributeValue(**"no"**));  
  
  
 liste\_langages.addContent(langage);  
}  
  
plex.addContent(liste\_langages);

Voici les résultats obtenus sur Firefox :



*Liste des projections*





*Détail d’un film*



*Détail d’un acteur*

# Petite conclusion

Le premier objectif ne nous a pas posé de problème. Nous avons pu le terminer assez rapidement. Toutefois, le second objectif nous a pris beaucoup plus de temps que ce que nous avions prévu. Nous devions également modifier notre structure XML afin qu’il corresponde au mieux au résultat désiré. Mis à part cet imprévu temporel, nous n’avons pas eu de difficulté à réaliser ce second objectif.

# Annexe

## Fichiers

* monschema.json : JSONSchema
* plex.xml : XML source (du labo 2)
* projections.xml : XML généré (projections.xml)
* structure.json : JSON (du labo 2)
* Amorce.java : Fichier servant à la création du fichier projection.xml