Master 1 Informatique

(Spécialité DL)

Unité d'enseignement DCLL

DOCUMENTATION TECHNIQUE

Projet :

Développement d'un convertisseur XML – JSON pour la plateforme Moodle

Equipe de développement

Kévin ANATOLE
Damien ARONDEL
Julien BARREAU
Cédric CAUSSE
Benjamin DUPUY
Quentin GROSSETTI

Rédigé le : 12/03/2013

Introduction

Ce document s'intègre dans le projet proposé par l'unité d'enseignement **DCLL** du Master 1 Informatique de l'Université Paul Sabatier. L'objectif est de développer un **convertisseur XML - JSON** pour la **plateforme Moodle**. Cette dernière présente le descriptif suivant :

- Plateforme de "e-learning" : https://moodle.org/.
- Propose un module de gestion de tests en ligne (**quizz**).
- Permet l'import et l'export d'une question ou d'un **quizz sous format XML** spécifié par la documentation Moodle : http://docs.moodle.org/22/en/Moodle_XML_format.

L'idée est donc de fournir un **programme écrit en Java** permettant de réaliser les tâches suivantes :

- Créer un fichier au **format JSON** à partir d'un fichier fourni au **format Moodle XML**, sachant que le fichier JSON résultat doit respecter la structure et le contenu du fichier initial.
- Créer un fichier au **format Moodle XML** à partir d'un fichier au **format JSON**, dont la structure et le contenu correspondent à un quizz Moodle.

Afin de prendre en compte ces exigences, le présent document détaille les différents points suivants :

- Analyse des deux **exigences liées au système** énoncées ci-dessus.
- Conception de l'**interface utilisateur** qui représente le programme principal.
- Dossier de **tests** à effectuer sur l'application.

I. Exigences liées au système

a. Librairie choisie

Pour plus de **simplicité** et au vu du **temps** qu'il nous est imparti, nous avons décidé d'utiliser une **librairie** créée par un autre groupe de développeurs. La communauté est plutôt active à ce propos et il nous a fallu en trouver une convenable, suivant les **critères** suivants :

- Propose une **conversion double** : de XML vers JSON et de JSON vers XML.
- **Réputation** parmi la communauté JSON Java.

Il faut savoir qu'en terme de conversion XML - JSON, rien n'est pour le moment **standardisé**. Certains problèmes surviennent, comme par exemple la **gestion des attributs** : JSON n'en a pas, XML en a. L'article suivant détaille très bien les différents cas de figure :

https://www.p6r.com/articles/2010/04/05/xml-to-json-and-back/

Nous avons toutefois essayé de trouver une **solution la plus adéquate** possible. Nous nous sommes tournés vers la librarie fournie par le **site officiel** de JSON (http://json.org/java/). Elle permet, à travers **15 classes Java**, de définir la conversion d'un côté comme de l'autre (XML vers JSON et vice versa).

b. Inconvénient et solution

La librairie évoquée ci-dessus présente néanmoins un **inconvénient majeur** : elle ne répond pas aux besoins de **conserver les attributs** (issue #1 sur l'article ci-dessus). A la place, elle les range comme éléments XML. Par exemple, pour la ligne XML suivante :

```
<question type="category">
```

Nous avons comme résultat JSON (cohérent) :

```
{"question": [

"type": "category"
```

Si on souhaite revenir en arrière et réaliser une **conversion dans l'autre sens** (JSON vers XML), le résultat sera quelque peu **différent** :

```
<question>
  - <type>category</type>
```

Pour régler ce souci, il suffit d'**analyser** un fichier Moodle XML type pour savoir où se situent **potentiellement** les attributs (sur **quels éléments** et avec **quel nom**). Nous avons donc 3 éléments contenant potentiellement, voire toujours, un **attribut** :

```
<question type="category">
  <answer fraction="100">
<questiontext format="moodle_auto_format">
```

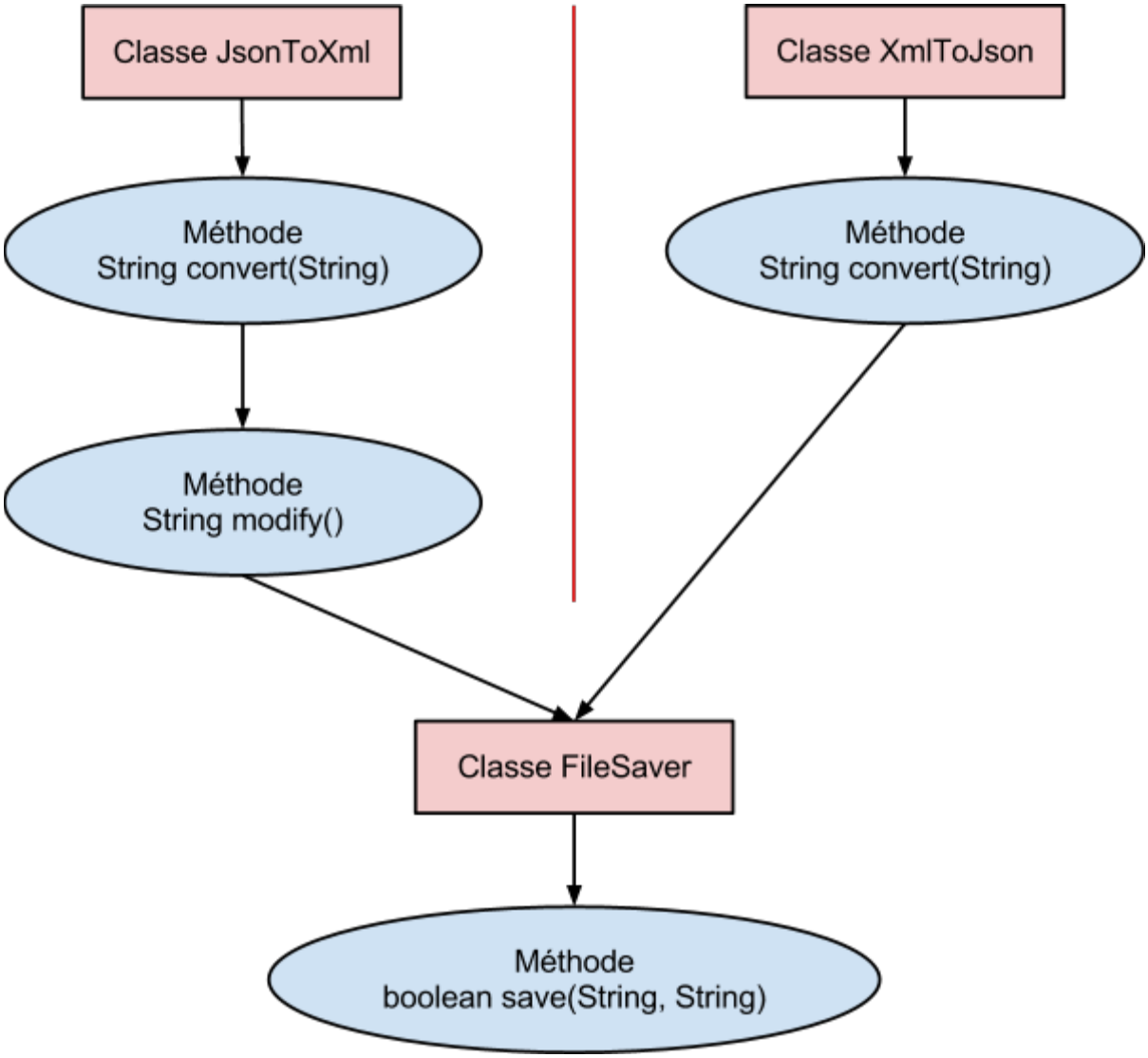
Sur le **fichier XML incorrect** (dû à la conversion JSON vers XML), l'idée est de faire **remonter** ces éléments **en attributs**. L'**algorithme utilisé** pour cela est le suivant (exemple pour l'élément <question>) :

Pour chaque noeud n du fichier XML incorrect **Faire**
    Si n.nom = "question" **Alors**
        Si n.contientSousNoeud("type") **Alors**
            n.ajouterAttribut("type", valeur("type"));
            n.supprimerNoeud("type");
        Fin Si;
    Fin Si;
Fin Pour;

Un **parseur XML** en Java est ainsi nécessaire pour parcourir le fichier d'une part et le modifier d'autre part. Nous nous sommes tournés vers **JDOM2** (http://jdom.org/) dans sa version 2.0.4.

c. Fonctionnement global

Le but de cette sous-partie est d'expliquer le **fonctionnement général** de l'application. Le **diagramme** suivant en dit plus à ce sujet :



Les méthodes **convert(String)** fournissent une chaîne de caractères représentant le contenu de ce qu'il résulte de la conversion. La méthode **modify()** permet de prendre en compte l'algorithme de la page 5. Enfin, la méthode **save(String, String)** permet de sauvegarder un fichier à un chemin donné avec un contenu. Elle renvoie un booléen qui indique si la sauvegarde a bien été effectuée.

II. Interface utilisateur

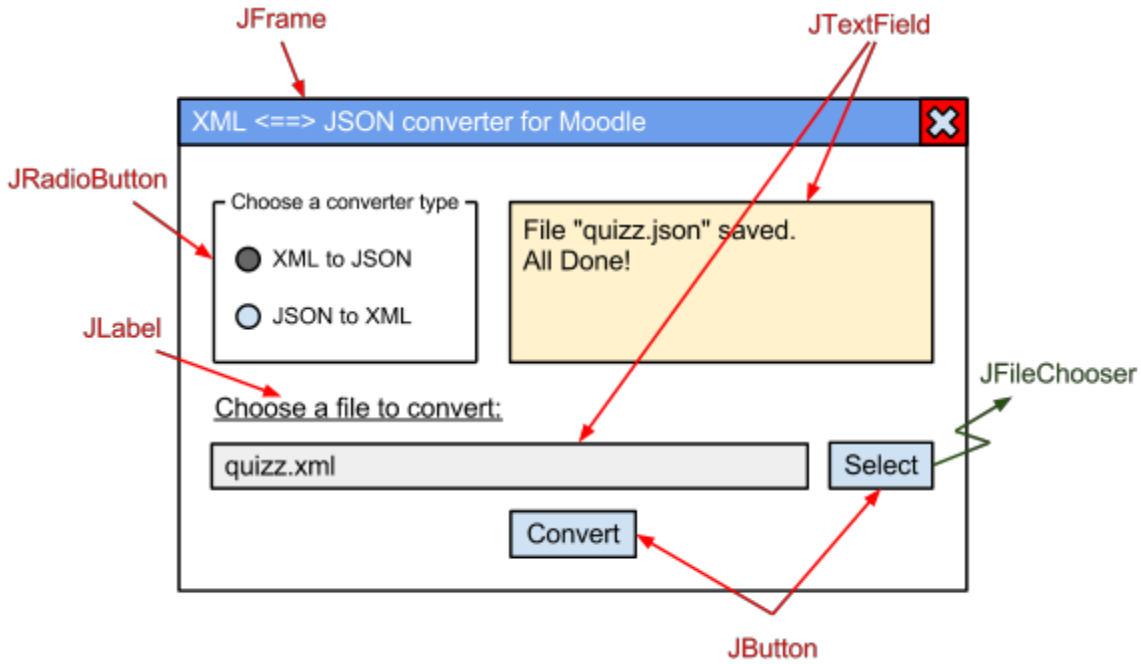
a. Choix de conception

Afin de rendre l'application plus **interactive**, la solution d'une **interface utilisateur** a prôné sur la mise en place du convertisseur sur un terminal type Linux. L'ensemble du programme étant **codé en Java**, nous avons décidé d'utiliser la **bibliothèque Swing**. Elle nous permettra de créer notre interface **en toute simplicité** tout en intégrant les exigences (voir partie I). Pour plus d'informations, nous vous invitons à consulter les sites suivants :

ttp://en.wikipedia.org/wiki/Swing_(Java) et http://docs.oracle.com/javase/tutorial/uiswing/

b. Maquette d'écran

Voici l'**interface utilisateu**r telle que nous l'avons conçu. Elle permet de **changer de type de convertisseur** facilement. De plus, un **message** (cadre jaune) permet d'indiquer à l'utilisateur si une **erreur** est survenue au cours du traitement ou non. La légende en rouge correspond aux noms des composants donnés par Swing.



c. Informations complémentaires

Le JTextField contenant “quizz.xml” dans la maquette d’écran est **grisé** tout au long de l’application, pour éviter les **erreurs potentielles** faites de l’utilisateur (faute de frappe, fichier de format incorrect). Quand l’utilisateur clique sur le bouton “Select”, le composant JFileChooser est appelé. Il est alors amené à choisir un fichier de **format conforme** au type de convertisseur choisi :

- Type “XML to JSON” (**type par défaut**) : fichier XML à choisir
- Type “JSON to XML” : fichier JSON à choisir

Dès que l’utilisateur a choisi, le champ grisé se met à jour avec le **chemin complet du fichier**. Le bouton “Convert” passe alors actif (**par défaut**, il est **inactif** s’il n’y a aucun fichier choisi). L’utilisateur n’a alors plus qu’à cliquer dessus pour effectuer la conversion d’un format à un autre. Un fichier est ainsi créé au **même chemin** que celui en entrée, avec le **même nom** mais avec une **extension différente**.

Voici un diagramme très simple montrant le **comportement de l’interface** (flèches rouges) en fonction des actions de l’utilisateur (flèches bleues) :