Bruno Quercia
Élève Ingénieur 3ème année
bruno.quercia@telecom-bretagne.eu

Rapport court de projet

P307 - Éditeur graphique de texte formaté

Année scolaire 2016 - 2017

Sommaire

1. CONTEXTE DU PROJET

Le projet P307 « Éditeur graphique de texte formaté » est présenté dans le cadre des projets S5 de Télécom Bretagne, en collaboration avec la société Openflexo.

Openflexo est une startup qui s'est donné pour mission de simplifier le travail des entreprises, soumises à une diversité croissante des types de documents échangés avec leurs clients, et confrontées à des difficultés d'harmonisation de ces documents.

Le projet Openflexo offre donc une solution logicielle Opensource permettant de mettre en lien les diverses données directement à partir des documents source. Contrairement à d'autres solutions qui consisteraient en une extraction et conversion des données dans un format commun, celle-ci permet de préserver les différents documents tels quels, et de les modifier en toute transparence.

Illustration de la méthode OpenFlexo (source : https://diatomee.openflexo.org)

Pour réaliser cette dernière fonctionnalité, l'application, développée en Java, fait appel à plusieurs bibliothèques, dont docx4all, qui fournit un outil de manipulation des fichiers Microsoft Word les plus récents (format docx).

2. REFORMULATION DU SUJET

L'adaptabilité à de nombreux types de documents (docx, odt, pdf, etc.) étant jusqu'ici réalisée par l'agrégation de plusieurs solutions « sur-mesure », dépendant d'autant de bibliothèques, l'application Openflexo en est devenue complexe, difficile à maintenir et à faire évoluer.

Le but du projet P307 est donc d'apporter une solution unique à la gestion de ces différents types de documents, via le développement d'un éditeur WYSIWYG (What You See Is What You Get) universel. Cet éditeur permettra de visualiser, modifier et sauvegarder les documents de l'utilisateur, qui seront gérés via une API (Application Programming Interface) générique.

La conception et l'implémentation de cette API constituent le véritable cœur du projet. Une première version de l'API a déjà été élaborée par l'équipe d'Openflexo sur la base empirique des structures rencontrées dans les bibliothèques utilisées, en particulier docx4all. Néanmoins, une version indépendante en est attendue afin de confronter les modèles pour en ôter tout biais.

L'API prendra la forme d'une interface, ou collection d'interfaces java, dont le rôle sera de décrire les attributs et méthodes essentiels d'un objet de type document générique.

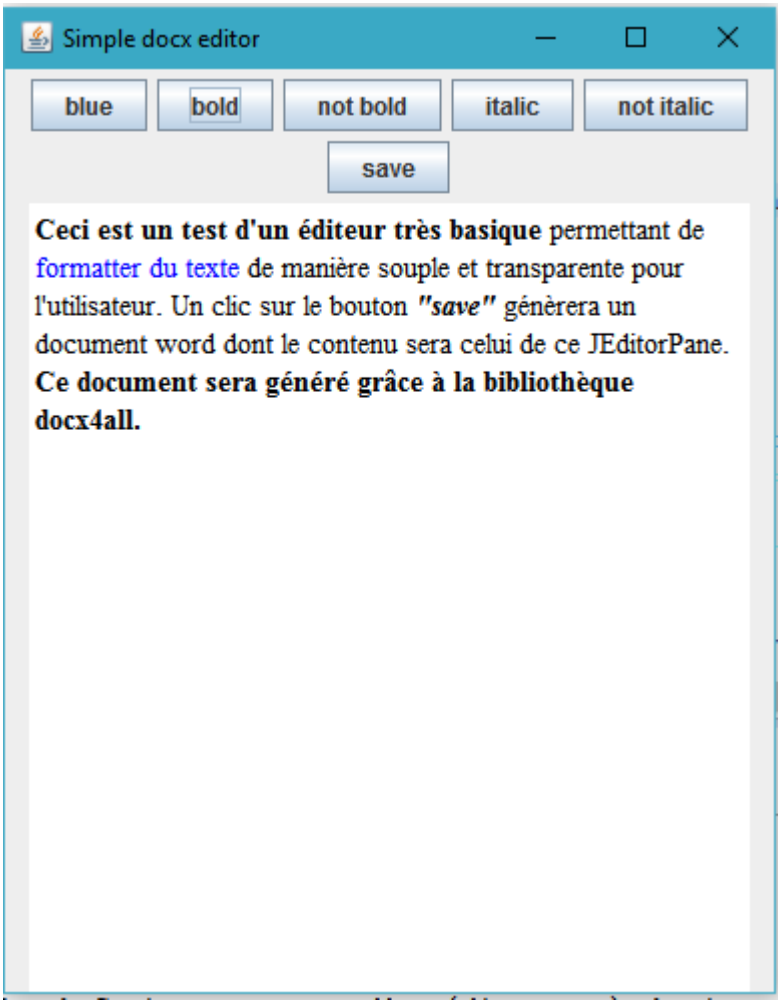
3. RAPPORT BIBLIOGRAPHIQUE

3.1 BIBLIOTHÈQUE GRAPHIQUE : SWING

L'application Openflexo utilise la bibliothèque Swing, qui est la bibliothèque graphique de référence Java, incluse dans le package Java Foundation Classes. Si elle est aujourd'hui supplantée par JavaFx, pensée pour étendre la gestion des applications mobiles, elle reste très utilisée et une forte compatibilité entre les deux bibliothèques est assurée. (Documentation Oracle, s.d.)

Parmi les composants de Swing, se trouve le composant JEditorPane, qui permet d'afficher du texte formaté. C'est donc ce composant qui devra être utilisé pour implémenter l'éditeur WYSIWYG.

Une première étude des interactions possibles avec ce composant a donné lieu à une petite application permettant de mettre en forme du texte au sein d'un JEditorPane, puis de le sauvegarder sous forme d'un document Word en utilisant la bibliothèque docx4all.



Premier essai à but auto-pédagogique

3.2 BIBLIOTHÈQUE DE MANIPULATION DE FICHIERS WORD : DOCX4ALL

À l'heure actuelle, l'application Openflexo utilise massivement la bibliothèque Opensource de docx4all, aussi appelée docx4j. Très complexe, cette bibliothèque offre toutes les fonctionnalités permettant de manipuler des fichiers de la suite Microsoft Office. (docx4java)

C'est cette complexité excessive au regard des fonctionnalités réellement utilisées par Openflexo, ainsi que l'inadéquation entre la modélisation spécifique aux documents de la suite Microsoft Office et le modèle de document générique voulu par la startup, qui la pousse à rechercher une solution se

passant de cette bibliothèque. Toutefois, cette dernière a servi et pourra encore servir de base à la réflexion tout au long de ce projet.

3.3 AUTRES BIBLIOTHÈQUES

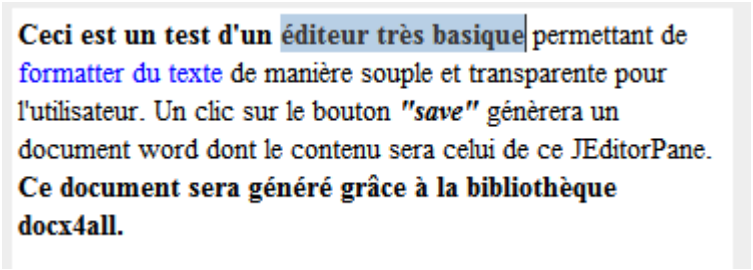
Afin de ne pas biaiser la réflexion et de construire une définition d'une généricité maximale d'un document, il sera judicieux d'étudier l'implémentation des mêmes fonctionnalités dans d'autres bibliothèques, parmi lesquelles jOpenDocument, qui est le pendant de docx4all pour les fichiers de type OpenDocument. Également OpenSource, elle s'inscrit parfaitement dans la ligne du projet Openflexo.

Apache PDFBox, une bibliothèque de manipulation de fichiers PDF, pourra également apporter un éclairage sur la structure d'un document. Son avantage principal est de s'intéresser à des documents tout autres (fichiers PDF) que les deux bibliothèques précitées.

4. ANALYSE DU PROBLÈME

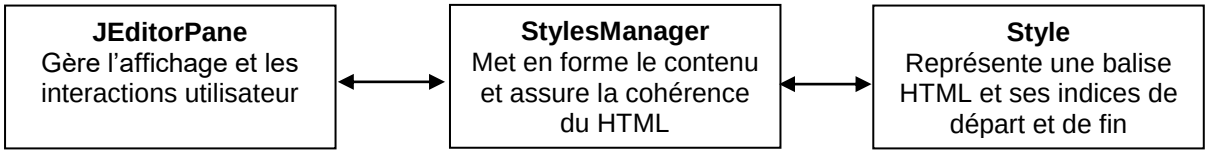
4.1 INTERFAÇAGE DES DIFFÉRENTS COMPOSANTS

Un JEditorPane utilise du HTML pour mettre en forme du texte. Ce HTML n'est pas directement modifiable par l'utilisateur : ce dernier peut certes insérer du contenu, mais les balises HTML doivent être gérées séparément par le programme. Ainsi, dans l'exemple suivant, l'utilisation de la fonction getSelectedText renvoie « éditeur très basique » et non « éditeur très basique » comme on pourrait s'y attendre.

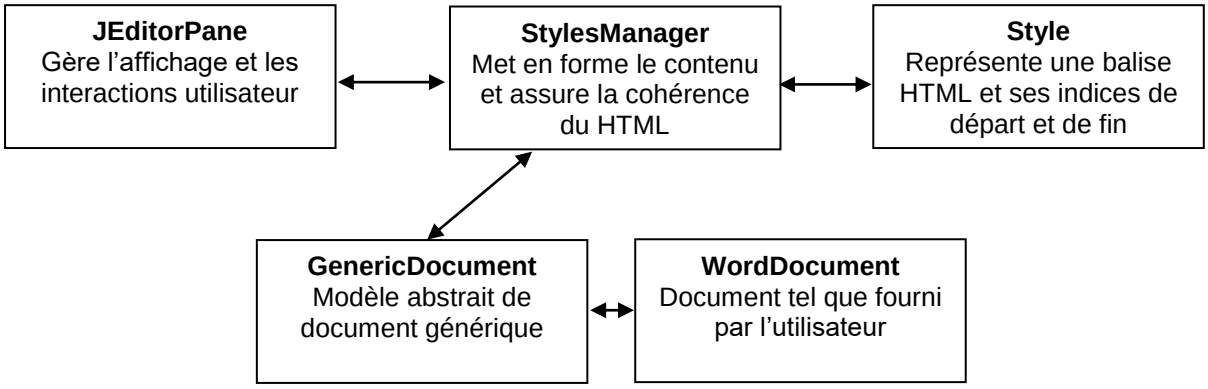


Exemple de sélection de texte dans un JEditorPane

De même, les fonctions permettant de modifier partiellement le texte prennent en argument du texte non formaté. Seule la fonction setText, dont le rôle est de définir l'intégralité du contenu du JEditorPane, permet d'utiliser des balises HTML. C'est pourquoi la structure de l'éditeur basique était la suivante :



À cette gestion de la mise en forme devra s'ajouter une correspondance avec le modèle de document générique. Le schéma sera donc complété de la manière suivante (exemple pour un document Word)



4.2 CONCEPTION DU MODÈLE

Le modèle est voulu le plus abstrait possible. Ceci implique de cerner les éléments fondamentaux constituant un document ainsi que les actions qui y sont associées, tout en anticipant des situations qui ne se produisent pas nécessairement dans les documents utilisés à l'heure actuelle, mais pouvant apparaître avec l'évolution des formats pris en charge par Openflexo.

Une illustration peut en être faite en reprenant l'exemple d'un document Word : à l'heure actuelle, les besoins d'Openflexo sont limités. Seul le texte sous ses différentes formes est exploité. Toutefois, il n'est pas à exclure que les documents présentés par l'utilisateurs soient plus riches, et comportent par exemple des objets OLE. Une première approche peut être d'ignorer simplement ces éléments, mais il paraît plus judicieux de les intégrer dans le modèle afin de prendre en compte leur existence et permettre, ultérieurement, une éventuelle évolution de l'application qui pourrait les prendre en charge.

Cela s'apparenterait à un type de contenu « autre », qui pourrait par la suite être raffiné entre autres par des héritages.

Une fois le modèle établi, il sera testé sur différents types de documents sur la base de tests unitaires, qui permettront de vérifier l'adéquation de rendu entre la source et le contenu présenté dans l'éditeur.

4.3 CODAGE DE L'ÉDITEUR

La finalité du projet restant le développement proprement dit de l'éditeur, il est nécessaire de prévoir une phase de codage, dont la fin marquera celle du projet. Toutefois, afin de réaliser les tests unitaires, il sera nécessaire d'avoir rapidement une version « basique » de l'éditeur, comportant *a minima* les fonctionnalité d'affichage principales. Ainsi, le développement devra se faire par étapes, au fur et à mesure des besoins imposés par la conception du modèle.

4.4 DOCUMENTATION

Destiné à être utilisé dans un projet Opensource évolutif, le travail devra être documenté – c'est d'ailleurs l'un des points fondamentaux évoqués dans sa description. Une description exhaustive des fonctionnalités devra être réalisée, ainsi qu'une explication sommaire des choix de structure effectués dans la conception du modèle de document générique.

5. PLAN DE TRAVAIL

Le projet se déroulera selon une méthode de développement agile, ce qui implique *a priori* de nombreux changements dans le calendrier prévisionnel. Ainsi, la version présentée dans ce rapport sert de guide initial, mais le déroulement réel pourra fortement différer. Compte tenu de la durée du projet et des fonctionnalités à réaliser, les réunions devront être non nécessairement longues mais fréquentes ; une réunion toutes les deux semaines paraît être un minimum absolu. Cela doit cependant rester une moyenne, car le temps imparti au projet n'est pas homogène, et pourra même être nul pendant plus d'une semaine. Lors de ces réunions, seront abordés les progrès réalisés et les difficultés rencontrées, puis les objectifs à court et éventuellement à long terme seront redéfinis en fonction de ce qui aura été évoqué.

5.1 OCTOBRE 2016 – MI-NOVEMBRE 2016 : COMPRÉHENSION DU PROBLÈME ET EXPÉRIMENTATION PRÉALABLE

Cette phase a d'ores et déjà été réalisée, et concrétisée par la réalisation de l'éditeur basique précité. Elle a également comporté une rapide étude de la bibliothèque docx4all, l'étude bibliographique évoquée dans le présent rapport, et une discussion des objectifs et du déroulement du projet avec l'équipe d'Openflexo. Elle prend fin par la remise de ce rapport.

5.2 MI-NOVEMBRE 2016 – DÉBUT DÉCEMBRE 2016 : ÉTABLISSEMENT DU MODÈLE

Au cours de cette phase, le modèle sera établi de façon théorique. La structure fondamentale d'un document ainsi que les actions inhérentes à cette structure seront analysées, et conceptualisées sous forme d'une représentation abstraite.

5.3 Décembre 2016 : Implémentation du modèle

Cette phase consistera à concrétiser le modèle par une interface ou collection d'interfaces Java, qui pourra dans un premier temps rester incomplète afin de procéder le plus rapidement possible aux tests.

Parallèlement à ce développement, une présentation du travail accompli et à accomplir sera réalisée, en vue de la soutenance orale du vendredi 16 décembre.

5.4 Janvier 2017 : Début du développement de l'éditeur et mise à l'épreuve du modèle Java

Une première version de l'éditeur sera réalisée afin de procéder aux tests unitaires de l'implémentation du modèle de document sur différents types de documents. Cette version fonctionnelle sera présentée lors de la Journée Portes Ouvertes de l'école en février 2017.

À l'issue de cette phase, le modèle théorique comme son implémentation devront avoir trouvé leur forme définitive.

5.5 Février 2017 : Codage de l'éditeur dans sa forme finale

L'éditeur, *a priori* limité à ce moment du projet à de simples fonctionnalités d'affichage, voire d'édition basique, devra se doter de l'ensemble de ses fonctionnalités finales dans son interface utilisateur. Ces fonctionnalités seront déterminées au cours des réunions en fonction de l'évolution du modèle de document générique.

5.6 Début mars 2017 : Finalisation du projet et rédaction du rapport

Cette phase consistera en l'aboutissement du projet, ainsi que la préparation de la soutenance finale et du rapport qui l'accompagne.

5.7 Cas particulier de la documentation

La rédaction de la documentation n'a pas de calendrier spécifique ; il semble souhaitable que celle-ci soit rédigée au fur et à mesure des implémentations. Elle pourra ainsi servir de guide dans l'évolution du projet ainsi que de base pour les bilans intermédiaires.

6. CONCLUSION

Ce projet nécessite une importante réflexion théorique ; si la partie codage n'est évidemment pas à négliger, elle ne peut se faire correctement sans l'établissement préalable d'un modèle complet de document abstrait. C'est pourquoi il est nécessaire de commencer au plus vite cette dernière. La méthode de développement agile permet de ne pas rester bloqué sur des obstacles théoriques qui repousseraient inévitablement les échéances, compromettant ainsi la réussite de l'ensemble du projet. L'organisation est donc primordiale.

Pour accélérer le développement, il pourra être intéressant de réfléchir un minimum aux solutions d'implémentation à mesure que la théorie sera développée.

7. BIBLIOGRAPHIE

Cette section est réduite, compte tenu de la part prépondérante de la création dans le cadre de ce projet.

(s.d.). Récupéré sur http://www.docx4java.org/trac/docx4j

Documentation Oracle. (s.d.). Récupéré sur http://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-overview.htm#JFXST784

www.telecom-bretagne.eu

Campus de Brest
Technopôle Brest-Iroise
CS 83818
29238 Brest Cedex 3
France
Tél. : + 33 (0)2 29 00 11 11
Fax : + 33 (0)2 29 00 10 00

Campus de Rennes
2, rue de la Châtaigneraie
CS 17607
35576 Cesson Sévigné Cedex
France
Tél. : + 33 (0)2 99 12 70 00
Fax : + 33 (0)2 99 12 70 19

Campus de Toulouse
10, avenue Edouard Belin
BP 44004
31028 Toulouse Cedex 04
France
Tél. : +33 (0)5 61 33 83 65
Fax : +33 (0)5 61 33 83 75

TELECOM Bretagne