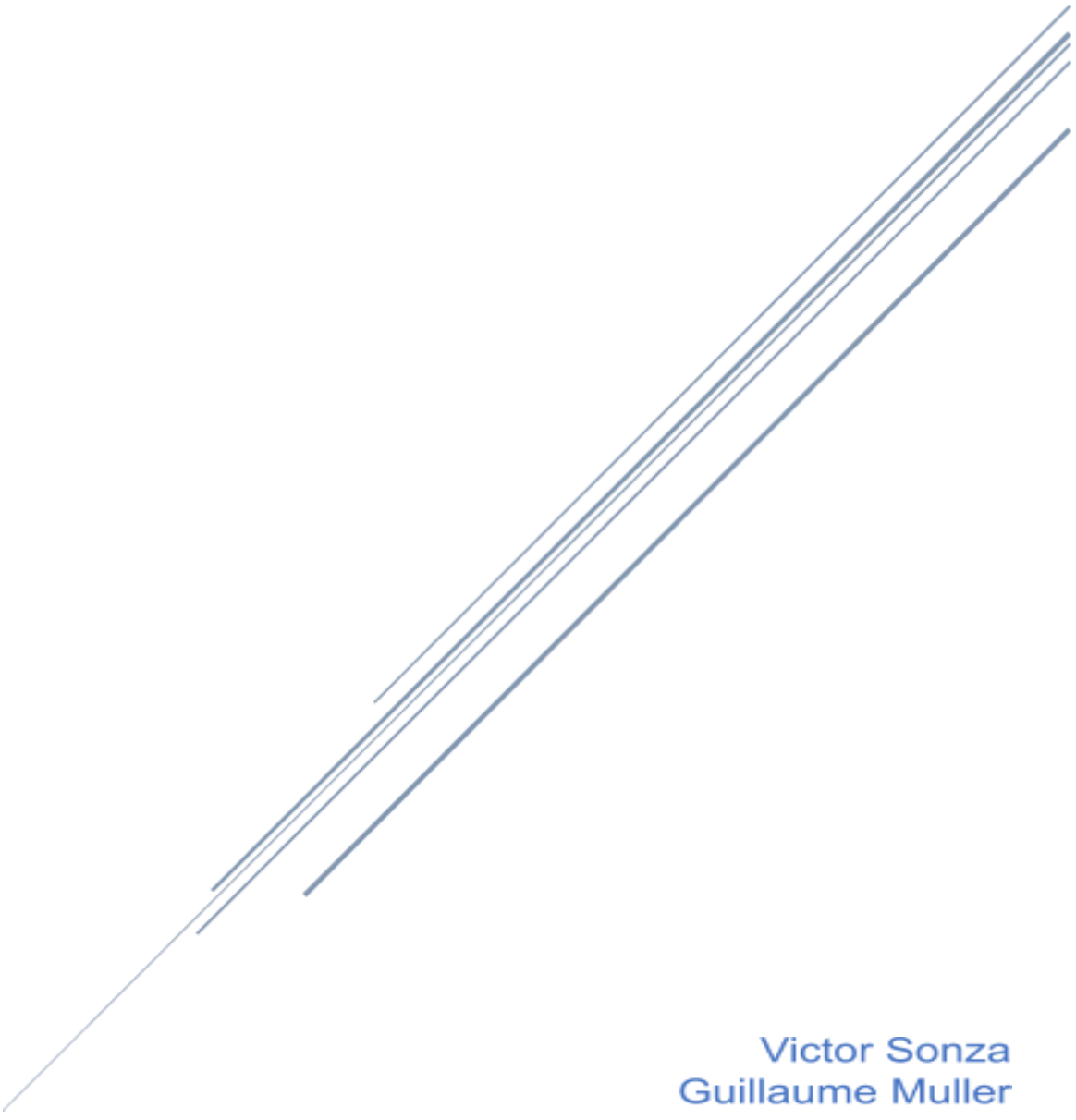
RAPPORT DE PROJET LO54

Gestion d'offre de formation



Victor Sonza
Guillaume Muller

I. Présentation de La technologie JasperReports

JasperReports est un outil de reporting open source. Il prend la forme d'une bibliothèque embarquable dans une application JAVA ou une application groovy. Cet outil permet de générer des documents à partir d'un ensemble de données venant de l'application.

JasperReports permet de générer des rapports dans différents formats :

- PDF
- HTML
- XLS/XLSX
- CSV
- RTF
- TXT
- SWF
- ODF

Les sources de données pour les rapports peuvent être :

- JDBC
- JavaBeans
- EJBQL
- XML
- Hibernate
- Fichier texte
- CSV

II. Tutoriel d'utilisation de JasperReports

a. Création du fichier JRXML

Dans un premier temps pour définir l'architecture du rapport il est nécessaire de créer un fichier au format XML contenant toutes les informations requises pour la mise en page de celui-ci.

Pour cela il est possible d'utiliser un éditeur graphique comme IReport indépendant de votre IDE, ou bien Jaspersoft Studio qui peut être intégré à Eclipse.

Le fichier peut aussi être édité et modifier à l'aide d'un éditeur de texte. L'extension de ce fichier est communément JRXML.

Ce fichier comprend d'une part la mise en page, et d'autre part le format des donnés et la requête SQL permettant de les récupérer.

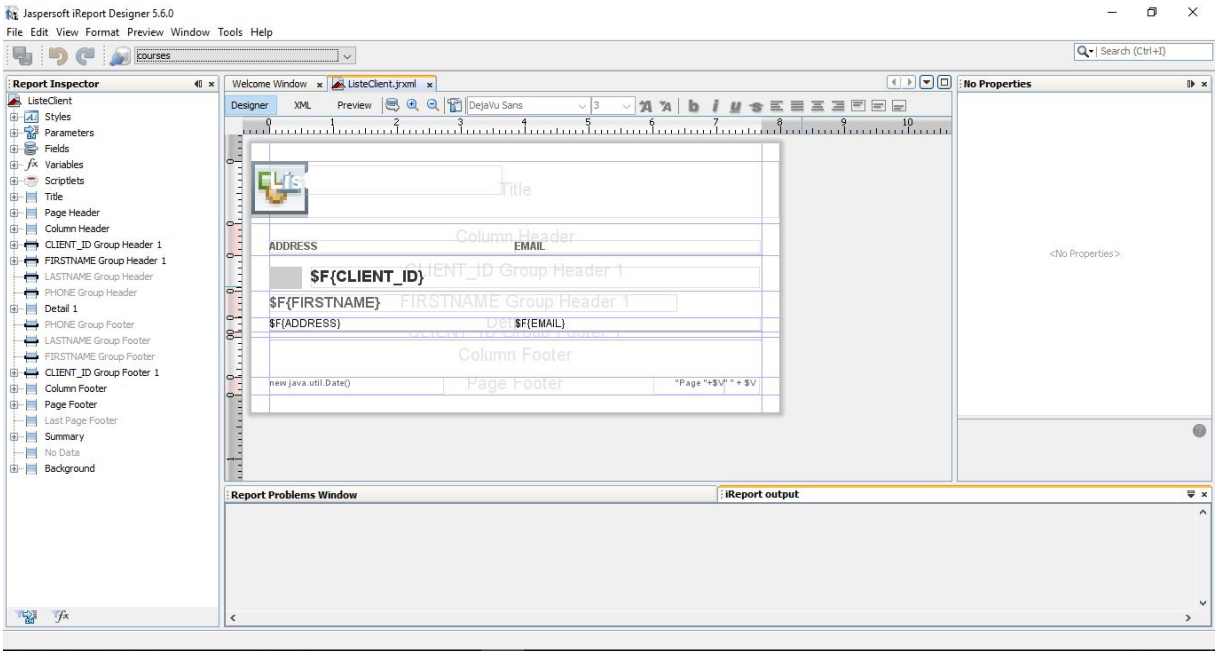
b. Utilisation d'IReport

IReport est un des éditeurs graphique permettant de créer les fichier JRXML. Il propose des Templates pour faciliter la création du rapport, mais il existe aussi un ensemble de possibilitées de personnalisation.

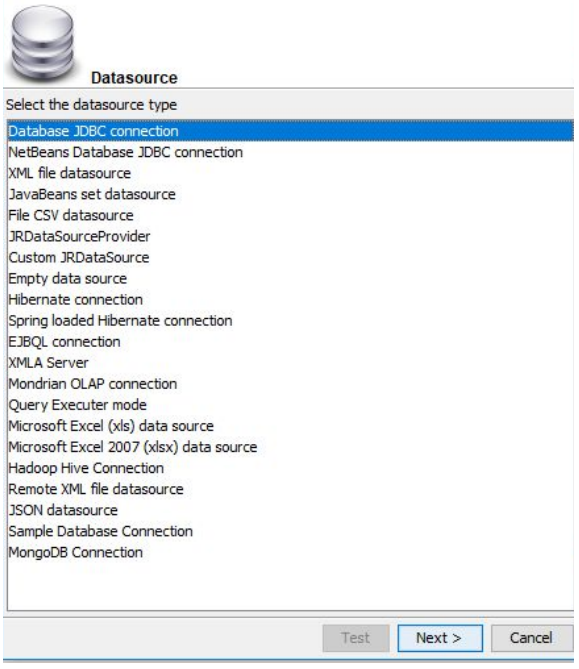
Pour créer un nouveau rapport il est nécessaire de connecter IReport à une source de donnée (une base de données, des JavaBeans (Collection), ...).

Quant IReport est connecté à la source de données, il est alors possible d'effectuer une requête SQL permettant de définir le format sous lequel les données seront récupérées. On pourra aussi éventuellement récupérer les données dans un même temps (prévisualisation du rapport).

IReport permet aussi de créer une version compilée du fichier JRXML qui est un fichier Jasper moins coûteux à charger pour une application.



IReport accepte de nombreuses sources de données et embarque un grand nombre de drivers de base de données (Oracle/PostgreSQL/...).



c. Création d'une classe permettant la génération du rapport

Dans un premier temps il est nécessaire de créer une classe permettant de gérer la connexion à la source de donnée du rapport (base de données, JavaBeans).

Cette classe importe le fichier JRXML ou Jasper, et génère le rapport au format spécifié dans ce fichier. Toutes ces opérations sont possible grâce aux fonction de la bibliothèque net.sf.jasperreports.

```
public class GeneratePDF {

    public  void ListeClient() {

        // - Paramètres de connexion à la base de données
        String url = "jdbc:derby://localhost:1527/courses";
        String login = "Courses";
        String password = "Courses";
        Connection connection = null;

        try {
            // - Connexion à la base
            Driver monDriver = new org.apache.derby.jdbc.ClientDriver();
            DriverManager.registerDriver(monDriver);
            connection = DriverManager.getConnection(url, login, password);

            // - Chargement et compilation du rapport
            JasperDesign jasperDesign = JRXmlLoader.load("C:\\Users\\guill\\Documents\\GitHub\\Gestion-offre-formation-\\LO54_Projet_Front_End\\src\\main\\resources\\jasperReportFile\\
            JasperReport jasperReport = JasperCompileManager.compileReport(jasperDesign);
            System.out.print("1");
            // - Paramètres à envoyer au rapport
            Map parameters = new HashMap();
            parameters.put("Titre", "Titre");
            System.out.print("2");

            // - Execution du rapport
            JasperPrint jasperPrint = JasperFillManager.fillReport(jasperReport, parameters, connection);
            System.out.print("3");
            // - Création du rapport au format PDF
            JasperExportManager.exportReportToPdfFile(jasperPrint, "C:\\Users\\guill\\Documents\\GitHub\\Gestion-offre-formation-\\LO54_Projet_Front_End\\src\\main\\resources\\jasperRep
        } catch (JRException e) {

            e.printStackTrace();
        } catch (SQLException e) {

            e.printStackTrace();
        } finally {
            try {
                connection.close();
            } catch (SQLException e) {

                    e.printStackTrace();
                }
        }

    }
```

Exemple de méthode de classe permettant la génération d'un rapport à partir d'une base de données derby et d'un fichier JXML

La classe et les méthodes publiques qu'elle possède sont alors utilisable dans l'application en important le package dans lequel elle se trouve (dans notre cas fr.utbm.lo54.front.lo54_projet_front_end.jasperReport).

III. Rapport d'expérience

a. Implémentation

Nous avons mis en place une relation directe entre notre module jasper report et notre base de données.

Nous aurions pu utiliser la couche d'accès aux données utilisée dans le reste de l'application (en utilisant par exemple des méthodes comme "GetAll") mais nous avons fait le choix technique de considérer la partie de génération des rapports comme un module indépendant du reste de l'application.

Nous avons généré les fichiers JRXML à l'aide du logiciel IReport.

Nous avons choisi d'utiliser les fichiers JRXML plutôt que que les version compilé pour des problème de chemin de fichiers. Les fichiers JRXML étant fondamentalement des fichiers XML, il est facile de modifier les chemins présents dans ceux-ci afin de permettre une adaptation aux déplacements des fichiers du projet.

Pour le format des rapports nous avons choisi le format PDF.

b. Difficultés

Durant la mise en place de JasperReports nous avons rencontré des difficultés:

Premièrement, un problème de version concernant Java pour l'installation d'IReport. Le fonctionnement de ce logiciel nécessite l'utilisation de Java SE 7 32bits car la maintenance d'IReport est limité aux problèmes critiques.

Ensuite, l'utilisation d'IReport nous a aussi posé quelques difficultés à cause de par notre utilisation du moteur de base de données Apache Derby (intégré à l'IDE Netbeans) dont le driver n'est pas intégré nativement dans IReport.

Enfin, la plus grosse difficulté que nous avons rencontré est liée au chemin des fichiers lors de la génération des rapports. Ces problèmes sont liés à la fois aux chemins internes des fichiers JRXML, ainsi qu'aux chemins présents dans la classe de génération. Nous avons donc été obligés d'utiliser des chemins absolu.

c. Améliorations possible

Pour améliorer l'application, nous pourrions notamment :

- Augmenter le nombre de format d'exportation
- Utiliser les fichiers compilés jasper afin d'alléger l'application
- Mieu gérer les chemins de fichier
- Permettre le téléchargement du rapport via le navigateur