

Reporting avec Jasper

David THIBAU – 2020

david.thibau@gmail.com



Agenda

- Introduction
 - Business Intelligence
 - Modèle OpenSource de la suite Jaspersoft
- La librairie *JasperReport*
 - Modèle de génération
 - Le fichier JRXML
 - Distribution et Exemples
- *JasperStudio* : Mise en route
 - Présentation / Installation
 - Sources de données
 - Dimension et bandes d'un rapport
 - Éléments simples
 - Paramètres, expressions, variables
- Éléments avancées
 - Groupes
 - Sous-rapports
 - Datasets, Listes et tableaux
 - Graphiques
 - Tableaux croisés
- Java et JasperReport
 - API de *JasperReport*
 - Scriptlets
- Annexes
 - Data Adapters
 - Gabarits et assistants
 - Report Books
 - Création programmatique de rapport



Introduction

Business Intelligence
Modèle Opensource et suite TIBCO JasperSoft



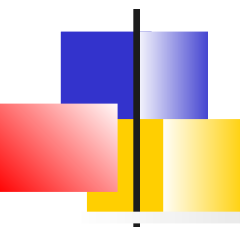
Business Intelligence

- La **Business Intelligence** (BI) : Technologies et applications qui permettent aux sociétés de mieux comprendre leur activité
=> proposer des historiques, des vues courantes ou prévisionnelles de l'activité de l'entreprise
- Les fonctions communes du BI sont :
 - Le data mining (L'extraction de données)
 - Le reporting
 - L'OLAP (Online Analytic Processing)
 - L'analyse de données (Agrégation)
 - L'analyse prévisionnelle
 - Le BigData, Le Machine Learning
- L'objectif étant l'aide à la décision.



Entrepôt de données

- Les outils et applications du BI manipulent de large volume de données.
On parle d'**entrepôts de données** (*datawarehouse*)
- Ces entrepôts sont construits grâce à des outils d'**ETL** (*Extract Transform Load* ou *datapumping*) qui extraient et transforment les données issues de plusieurs sources hétérogènes
 - A un degré de moindre, les activités de reporting utilisent les **bases de données relationnelles de production**.



Modèle OpenSource et suite JasperSoft



Acteurs du marché

- Les acteurs traditionnels du marché datant des années 1970 ont souvent été intégrés dans les géants de l'informatique :
 - Business Objects → SAP
 - Cognos → IBM
 - Hyperion → Oracle
 - SAS
 - Microstrategy
 - Information Builders
- Ces acteurs sont désormais concurrencés par des solutions plus légères et moins onéreuses, basées généralement sur des modèles OpenSource : La suite Jaspersoft, L'outil Birt



TIBCO

La société TIBCO a racheté les projets *JasperReport* afin de proposer une offre OpenSource Commerciale autour du BI : **La suite JasperSoft**

Pour les composants de la suite, 2 éditions sont proposées

- **Edition communautaire** : Entièrement gratuite, elle profite de la communauté pour contribuer et tester les nouvelles fonctionnalités
- **Edition commerciale** : Offre des versions plus testées, donne accès au support et à un engagement légal de TIBCO (certification) , offre certaines fonctionnalités supplémentaires



Partenariat

- *TIBCO Jaspersoft* développe également des partenariats avec les intégrateurs ainsi qu'avec d'autres éditeurs
 - Il certifie les intégrateurs qui proposent donc sa solution à leurs clients
 - Il est partenaire d'un certain nombre d'éditeurs connexes : Serveur Java EE, BD, BigData ...
 - Il s'intègre dans d'autres produits (Novell, Salesforce.com)



Objectifs de la suite *Jaspersoft*

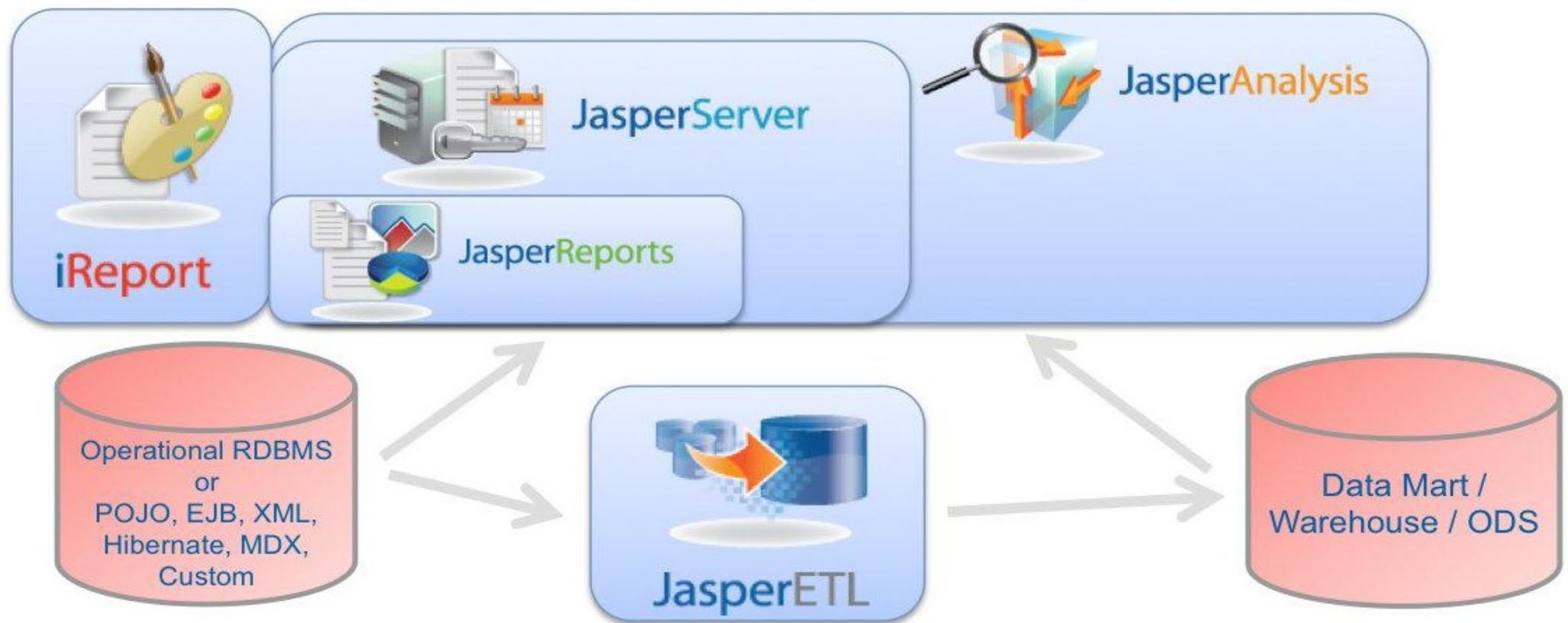
- Les objectifs de TIBCO avec la suite Jaspersoft
 - Une suite BI complète prête à l'emploi avec offre Cloud
 - Facilement intégrable dans des applications métier
 - Fonctionnelle mais abordable en terme de complexité
 - Compétitive en terme de coût de déploiement
 - Interopérable avec des systèmes existants
 - Viralisation via un mode OpenSource
 - Support des devices mobiles, des technologies Javascript



Les différents composants de la suite JasperSoft

- **JasperReport** :
Bibliothèque Java au cœur de la suite capable de générer des rapports en différents formats à partir d'une description XML.
- **IReport / JasperStudio** :
Outil graphique WYSIWYG permettant l'élaboration de rapports destinés aux développeurs comme aux fonctionnels métier.
- **JasperServer** :
Serveur de publication de rapports destinés aux fonctionnels métier. Déploiement de rapport, Droits d'accès, Programmation de tâches de génération, Envoi de rapports, Interface REST.
- **JasperAnalysis** :
Application d'analyse de données OLAP interactive destinée aux fonctionnels métier.
- **JasperETL** :
Outil d'intégration de données basé sur Talend.

Suite Jaspersoft





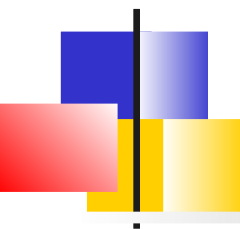
Modes de licences

- 3 types de licences :
 - **JasperSoft OEM** destiné aux revendeurs ou aux intégrateurs des produits JasperSoft dans les applications qu'ils réalisent
 - **Commerciale** : Destiné aux utilisateurs finaux des produits JasperSoft, en particulier *jasperStudio*, *JasperServer* et *JasperAnalysis*
 - **Licence OpenSource**
 - **GPL** : Licence contaminante, l'application contenant le produit JasperSoft doit être également OpenSource
 - **LGPL** : Licence non-contaminante, l'application contenant le produit JasperSoft peut être commerciale



Comparatif

	Communautaire	Commerciale / Cloud
JasperReport	LGPL	Identique
JasperStudio	GPL	Palette augmentée
JasperServer	GPL	Plus métier, Multiple Sources de données, Cache mémoire ; BigData,
JasperAnalysis	GPL	Quelques Add-ons



JasperReport

Modèle de génération
Le fichier JRXML
Distribution et exemples



Introduction

- *JasperReport* est une librairie entièrement Java .
- Disponible sous forme de jar, elle peut être intégrée à toute application Java :
 - Application desktop
 - Application web
- La librairie est capable de générer des rapports à partir d'un design au format XML et d'une source de données (base relationnelle ou autre)
- Le rapport généré peut être dirigé vers différents media :
 - Ecran
 - Imprimante
 - Fichiers (PDF, HTML, XLS, RTF, ODT, CSV, TXT et XML)



Modèle de génération

La génération du rapport s'effectue en 4 étapes principales :

1. Mise au point d'un fichier **JRXML** définissant

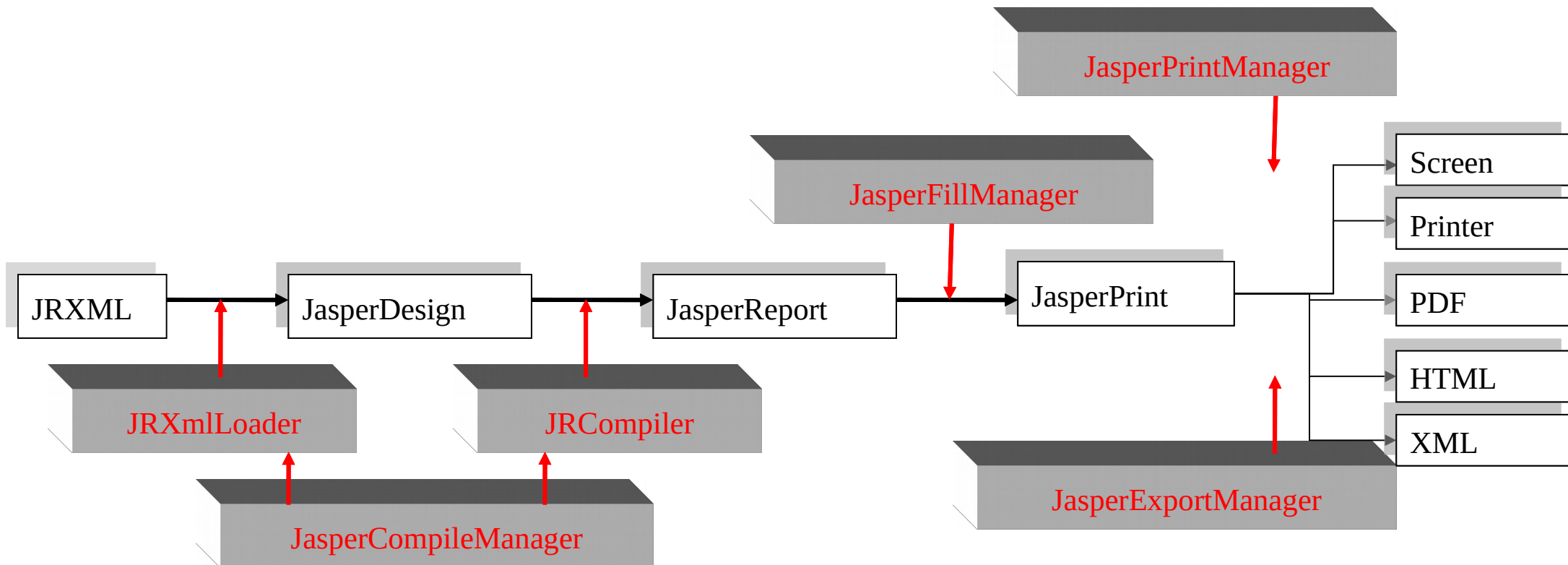
- La structure du rapport
- Sa disposition et son rendu visuel
- Les données dynamiques provenant d'une source de données (Requête, calcul, ...).

2. **Compilation** du rapport. Résultat : Fichier binaire *.jasper*

3. **Fusion** : Production du rapport avec récupération dynamique des données à partir de la source de données. Résultat : Fichier binaire *.jrprint*

4. **Exportation** du rapport dans un format visualisable par un viewer

Cycle de vie d'un rapport





Usage

- 1) Les développeurs mettent au point un rapport dans leur environnement en utilisant *JasperStudio*
- 2) Le rapport compilé est déployé dans *JasperServer* ou dans une application métier intégrant la librairie *JasperReport*
- 3) Les rapports sont générés dynamiquement par les utilisateurs finaux ou des traitements batch



Distribution



Installation

- Pré-requis :
 - Java installé
 - Outil *ant/ivy* permettant d'exécuter les exemples et de générer la documentation (utilisation de *hsqldb*)
- Téléchargement de la distribution au format zip ou tar.gz
(<http://sourceforge.net/projects/jasperreports/>)
- Possibilité de consulter les sources et de construire la librairie à partir des sources via scripts *ant*



Distribution

- La distribution contient les sources du projet et des fichiers de build *Ant/Ivy* et *Maven* permettant de construire la distribution et la documentation et d'exécuter les exemples
 - **src** : Sources Java
 - **build** : Répertoire de compilation
 - **demo** : Exemples
 - **doc** : Sources pour la documentation (configuration et schéma)
 - **lib** : Librairies tierces
 - **dist** : Génération de la distribution

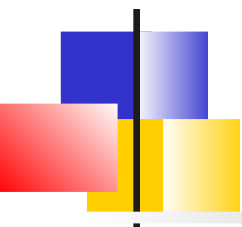


Configuration du moteur

Lors de la génération, le moteur *JasperReport* lit sa configuration à partir d'un fichier

jasperreport.properties

- De nombreuses options de configuration sont disponibles.
- La documentation peut être consultée dans
dist/docs/config-reference.html



Le fichier JRXML



Rappels sur XML

- Bien formé :
 - 1 seul élément racine : (*<jasperReport>*)
 - Pas de balises imbriquées
 - Attributs entre guillemets
- Référence à un schéma décrivant la syntaxe :
 - Version 3.6 :
<http://jasperreports.sourceforge.net/xsd/jasperreport.xsd>
- Documentation :
<dist/docs/schema.reference.html>



Structure du fichier

Sous la balise racine, on trouve séquentiellement :

- Des éléments **déclaratifs**
- Des éléments **structurels** définissant des bandes horizontales d'une certaine hauteur
- A l'intérieur des bandes, des **composants d'affichage** issus de la palette



Éléments déclaratifs

- ***property**** : N propriétés arbitraires globales au rapport
- ***import**** : N importation de packages java
- ***template**** : N références à des gabarits de styles
- ***reportFont**** : N définitions de polices
- ***style**** : N définitions de styles
- ***subDataset**** : N références à des jeu de données (exemple : requêtes SQL)
- ***scriptlet**** : N références à des scriptlets (Code spécifique Java)
- ***parameter**** : N définitions de paramètres de rapport
- ***queryString?*** : 1 seul requête principale (SQL ou autre)
- ***field**** : N définition de champs
- ***sortField**** : N champs de tri
- ***variable**** : N définitions de variable
- ***filterExpression?*** : 1 expression booléenne pour filtrer les données
- ***group**** : N définition de regroupement de données



Éléments structurels

- ***background?*** : 1 fond de page
- ***title?*** : 1 bande de titre, affichée une seule fois en début de rapport
- ***pageHeader?*** : 1 entête de page affichée à chaque début de page
- ***columnHeader?*** : 1 entête de colonne (pour les rapports multi-colonnes) affichée à chaque changement de colonne
- ***detail?*** : 1 bande détail. Affichée autant de fois qu'il y a de résultat dans la requête principale
- ***columnFooter?*** : 1 pied de colonne (pour les rapports multi-colonnes)
- ***pageFooter?*** : 1 pied de page
- ***lastPageFooter?*** : Le pied de page de la dernière page
- ***summary?*** : 1 résumé affiché en fin de document
- ***noData?*** : Texte alternatif quand il n'y a pas de données



Exemple

```
<?xml version="1.0" encoding="UTF-8"?>
<jasperReport
  xmlns="http://jasperreports.sourceforge.net/jasperreports" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://jasperreports.sourceforge.net/jasperreports
http://jasperreports.sourceforge.net/xsd/jasperreport.xsd
  name="report2" pageWidth="612" pageHeight="792" leftMargin="20" rightMargin="20" topMargin="20" bottomMargin="20">
  <background> <band splitType="Stretch"/> </background>
  <title>
    <band height="79" splitType="Stretch">
      <staticText> <reportElement x="236" y="29" width="100" height="20"/>
        <text><![CDATA[Mon titre centré]]></text>
      </staticText>
    </band>
  </title>
  <pageHeader> <band height="35" splitType="Stretch"/> </pageHeader>
  <detail> <band height="125" splitType="Stretch"/> </detail>
  <pageFooter>
    <band height="54" splitType="Stretch">
      <textField pattern="EEEE dd MMMM yyyy"> <reportElement x="472" y="34" width="100" height="20"/>
        <textFieldExpression class="java.util.Date">
          <![CDATA[new java.util.Date()]]></textFieldExpression>
        </textField>
      <image> <reportElement x="0" y="46" width="78" height="8"/>
        <imageExpression class="java.lang.String">
          <![CDATA["/home/Jasper/images/logo-plb-footer.gif"]]></imageExpression>
        </image>
      </band>
    </pageFooter>
  </jasperReport>
```



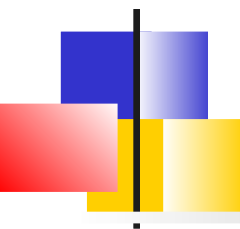
Jasper Studio Mise en route

Présentation et installation
Source de données et Champs
Dimensions et bandes d'un rapport
Éléments simples
Paramètres, expressions, variables



Introduction

- *jasperStudio* est un outil de design **WYSWIG** qui propose un environnement complet pour créer, tester, prévisualiser ou exporter les rapports JasperReport (S'appuie sur Eclipse)
- Les sources de données peuvent provenir de **toutes les sources supportées par JasperReport** : bases de données, connections OLAP XML/A, fichiers XML ou CSV collections de JavaBeans et même les sources spécifiques
- Il supporte **différents langages de requête** comme SQL, HQL, XPath, EJBQL et MDX et permet même de plugger d'autres langages si nécessaire (PL/SQL).
- Pour les développeurs, *jasperStudio* peut **générer les fichiers .jasper** qui peuvent être ensuite déployés dans les applications java
- *jasperStudio* peut également être utilisé pour déployer les rapports directement dans **JasperServer**.



Principales fonctionnalités

- Compatibilité complète avec les balises XML JasperReports ... si utilisation des bonnes versions
- Éditeur WYSIWYG, outils de dessin (rectangles, lignes, ellipses, labels, graphiques, sous-rapport, tableau croisé),
- Coloration syntaxique pour les expressions (Java, Javascript, Groovy)
- Support pour tous les langages Unicode (Russe, Chinois, Japonais, etc.). Support pour l'internationalisation
- Navigateur de la structure du document : *Outline*
- Outils de compilation, fusion, exportation et prévisualisation intégrés
- Assistants pour créer les rapports, les sous-rapport, les tableaux croisés, les graphiques
- Support pour les gabarits de documents, les polices TrueType. Bibliothèques de Styles.
- Bibliothèque d'objets standards (par exemple le nombre de pages)
- Fonctionnalité Drag-and-drop, Undo/redo illimité
- Explorateur d'un repository *JasperServer*.
- Support de toutes les bases de données accessibles par JDBC, extensibles à d'autres types de sources de données
- Assistant de mise au point de requêtes SQL, MDX, XPath



Documentation disponible

- Site web
<https://community.jaspersoft.com/documentation?version=49176>
- En particulier :
« *JasperStudio User Guide* »



Installation

- Pré-requis :
 - Java 1.8+
 - 256MB RAM disponible + 50MB d'espace libre
- Distributions binaires disponibles :
 - *TIBCOJaspersoftStudio-x.x.x.tgz* : Binaire au format TAR GZ pour une distribution Linux
 - *TIBCOJaspersoftStudio-x.x.x-windows-installer.exe* : Installeur pour Windows.
- Également disponible sous forme de plugin pour l'environnement de développement *Eclipse*



Installation

- A partir de la version binaire, il suffit de la décompresser, puis de lancer l'exécutable : *Jasper Studio*
 - Avec certaines releases d'Ubuntu utiliser le script `./runubuntu.sh` (Ubuntu)
- L'installateur Windows, ajoute une entrée dans « Mes programmes » + un raccourci sur le bureau

Présentation *JasperStudio*

The screenshot displays the JasperStudio IDE interface. The central workspace shows a report design for 'Cherry Title' with a subtitle 'Cherry SubTitle'. The report includes a table with columns: ADDRESS_ID, ADDRESS_FIRSTNAME, ADDRESS_LASTNAME, and ADDRESS_STREET. The table contains data rows with values like 'new java.util.Date()', '\$F{ADDRESS_CITY}', '\$F{ADDRESS_ID}', '\$F', '\$F', '\$F', and 'Page: ~SV~ * ~SV~'. The left sidebar contains the 'Repository Explorer' and 'Project Explorer' panels. The 'Project Explorer' shows a project named 'FirstReport' with various files and folders. The right sidebar contains the 'Palette' panel with elements like Note, Champ de Texte, Texte Statique, Image, Saut, and Rectangle. The 'Properties' panel shows the 'Report: FirstReport' properties, including 'Nom Rapport', 'Langage', 'Importations', 'Classe Usine de Format', 'Si Pas de Type de Données', and 'Dataset'. The 'Report State' panel at the bottom shows the 'Console' with compilation and execution statistics.

Report Design:

Cherry Title
Cherry SubTitle

ADDRESS_ID	ADDRESS_FIRSTNAME	ADDRESS_LASTNAME	ADDRESS_STREET
new java.util.Date()			
\$F{ADDRESS_CITY}			
\$F{ADDRESS_ID}	\$F	\$F	\$F
new java.util.Date()			Page: ~SV~ * ~SV~

Report State:

Metric	Value
Compilation Time	0,387 sec
Filling Time	1,293 sec
Report Execution Time	1,833 sec
Export Time	0 sec
Total Pages	3 pages
Processed Records Count	50 records
Fill Size	0 bytes



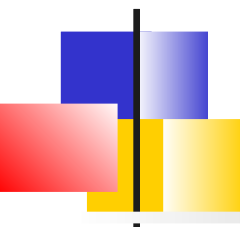
Multi-fenêtrage

- L'espace de travail de *jasperStudio* est multi-fenêtre :
 - **Espace de design central** permettant d'éditer les rapports, de visualiser le XML et de prévisualiser
 - **Outline** : Vue hiérarchique du rapport, permettant de sélectionner ses éléments
 - **Vue propriété** : Permet d'éditer les propriétés de l'élément sélectionné
 - **Palette** : Placer des éléments sur le rapport
 - **Outils de positionnement** : Alignement des éléments
 - **Problems** : Avertissements, Problème de parsing XML, de positionnement
 - **Report state** : Résultat des programmes internes (Compilation, fusion de données, ..)
 - **Explorateur du repository** JasperServer
- Accès à la configuration générale via
Windows → Preferences → JasperSoft Studio



Test de la configuration

- Tester la configuration en créant un rapport vide :
 1. *File → New → Jasper Studio → JasperReports Project*
 2. *File → New → JasperReport*
 3. *Choix du gabarit « blank »*
 4. *Choix du nom de fichier.*
- Si tout est normal, *jasperStudio* génère un fichier *.jasper* et affiche une prévisualisation d'une page blanche



Sources de données



Introduction

- *JasperReport* supporte différents types de source de données ou **Data Adapter** (via son interface Java *JRDataSource*)
- L'implémentation la plus utilisée est une base de données relationnelles via JDBC (implémentation *JRResultSetDataSource* qui encapsule un objet *java.sql.ResultSet*)



Itération *JRDataSource*

- Une source est tout simplement un objet capable d'itérer sur un ensemble d'enregistrements : les lignes.
- C'est un objet ***consommable*** : Lors de la génération, les lignes sont lues une à une et il n'est pas possible de revenir en arrière



Illustration *JRDataSource*

A tout moment de la génération, il y a un enregistrement courant

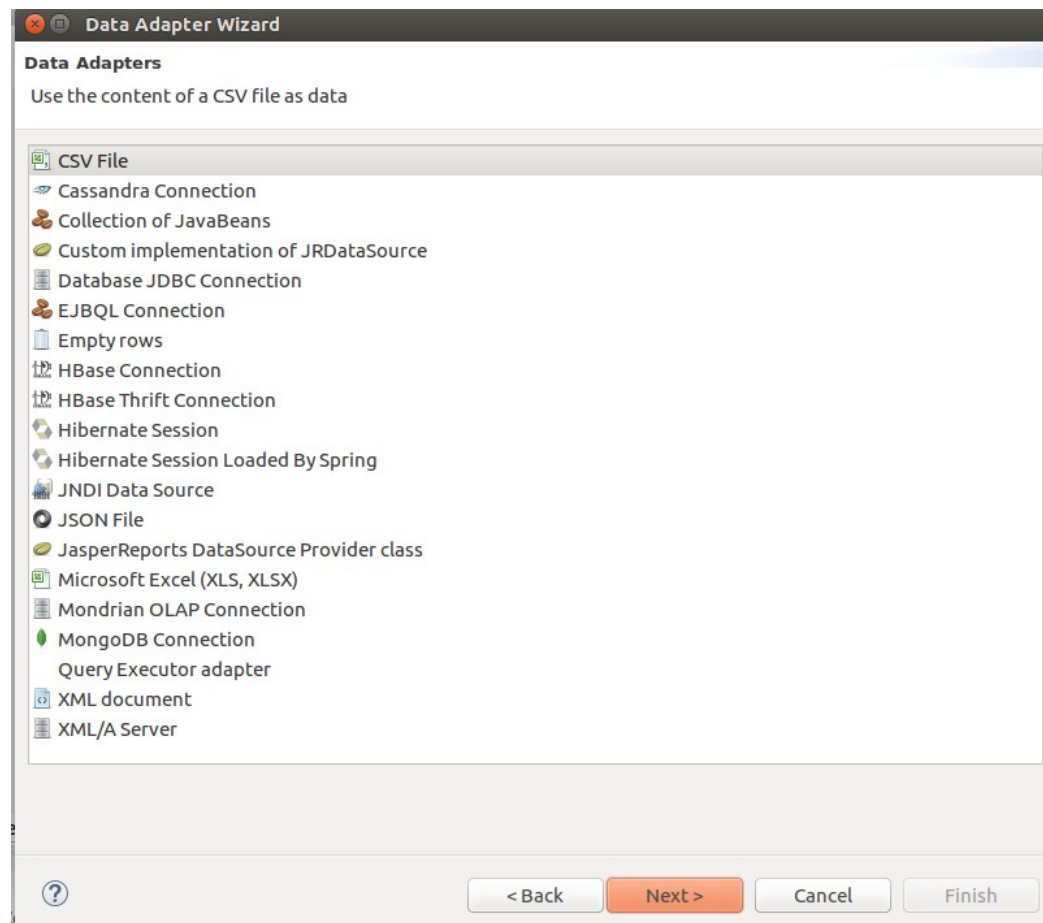
1	36001	36140	Aigurande	ut
2	36002	36150	Aize	ut
3	36003	36120	Ambrault	ut
4	36004	36210	Anjouin	utvatan
5	36005	36120	Ardentes	utvatan
6	36006	36200	Argenton-sur-Creuse	utlachatre
7	36007	36500	Argy	utleblanc
8	36008	36700	Arpheuilles	utleblanc
9	36009	36330	Arthon	utvatan
10	36010	36290	Azay-le-Ferron	utleblanc
11	36011	36210	Bagneux	utvatan
12	36012	36270	Baraize	utlachatre
13	36013	36110	Baudres	utvatan



Data Adapter supportés

- **JDBC connection / JNDI Data Source**: Connexion à une base de données relationnelles en Java
- **Code Java** : Collection de JavaBeans, Hibernate, EJB, Implémentation custom de *JRDataSource*
- **Fichier simple** : XML, CSV, JSON, Excel
- **NOSQL** : Cassandra, MongoDB
- **Connexion OLAP Mondrian ou XMLA** : Connexion utilisé pour l'analyse de données
- **Source de données vide** : Le rapport n'a pas de données dynamique

File → New Data Adapter





Rappels sur JDBC

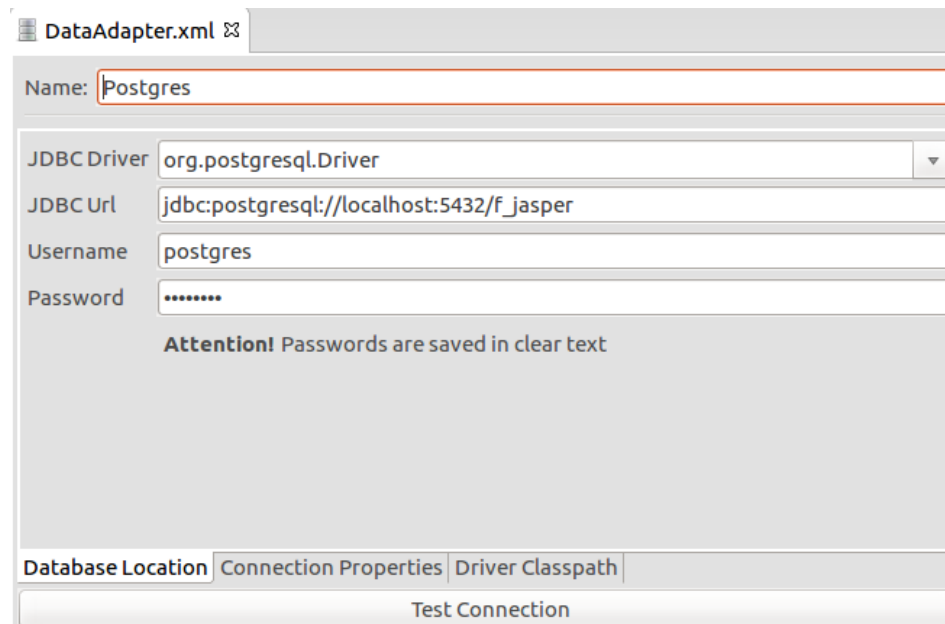
- Une connexion JDBC permet à un programme Java de se connecter à une base de données relationnelles (toutes les bases sont supportées)
- Pour cela, il est nécessaire d'utiliser les pilotes JDBC (bibliothèque Java) fournis par l'éditeur de base.
- Configurer une connexion JDBC consiste à fournir :
 - La **classe principale** du Driver
 - L'**URL** d'accès à la base (syntaxe dépendante de l'éditeur)
 - Un **compte** base de données valide (login/mot de passe)
- *jasperStudio* fournit les drivers JDBC pour *HyperSonic*. Les autres doivent être installés



Installation Driver

La bibliothèque du driver JDBC doit être trouvée par *jasperStudio*.

Lors de la création d'un DataAdapter, on lui indique où se trouve le .jar contenant le Driver



DataAdapter.xml

Name: Postgres

JDBC Driver: org.postgresql.Driver

JDBC Url: jdbc:postgresql://localhost:5432/f_jasper

Username: postgres

Password:

Attention! Passwords are saved in clear text

Database Location | Connection Properties | Driver Classpath

Test Connection



Dissociation rapport et source de données

- Il n'y a pas d'association stricte entre un rapport et une source de données.
- Lors de la génération, la source de données est fournie en paramètre
- Il est ainsi possible de réutiliser le même design sur différentes sources.
 - Exemple : Base de données de développement et base de données de production



Erreurs communes

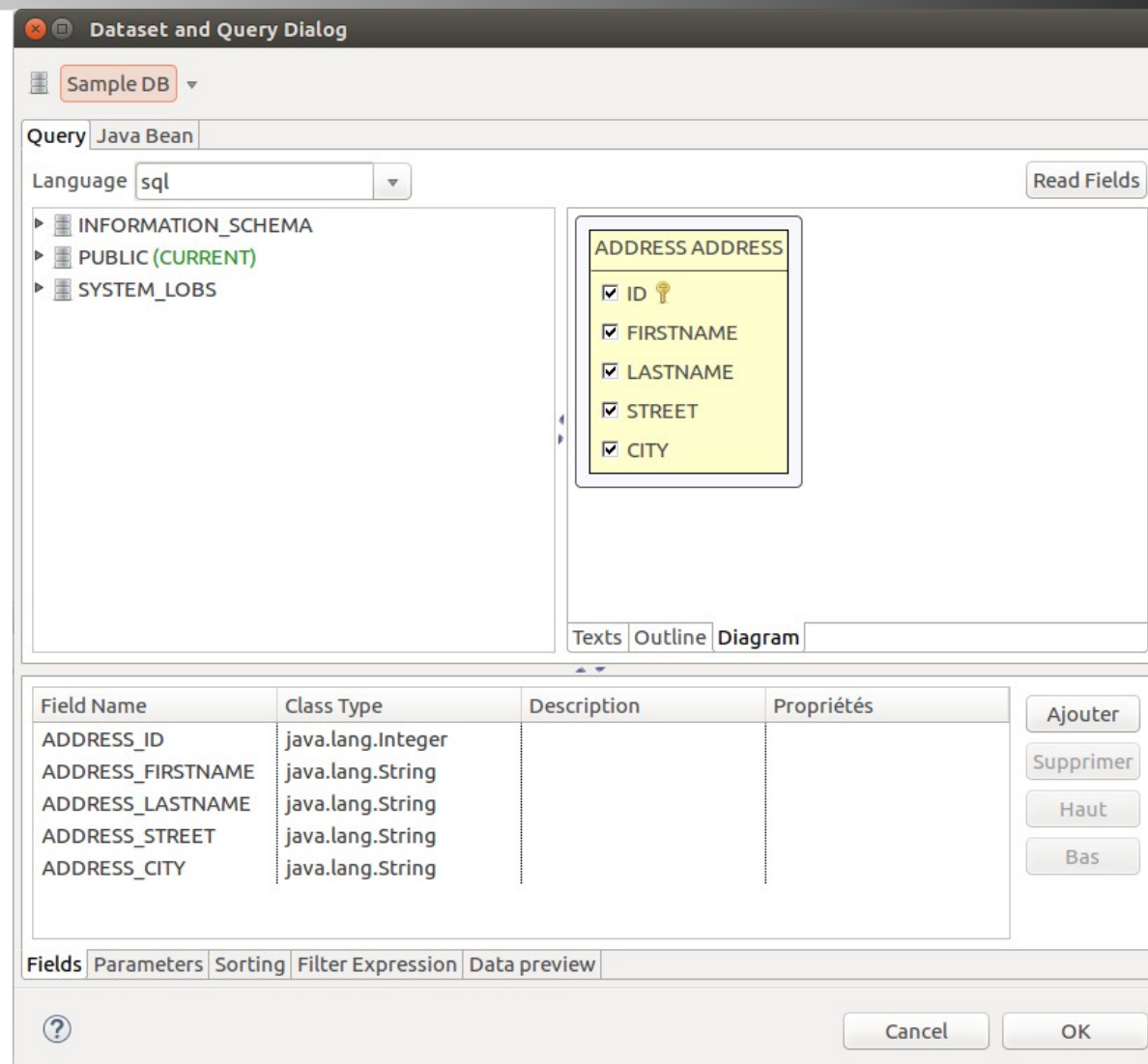
- 3 types d'erreurs peuvent survenir lorsque l'on configure une connexion JDBC :
 - ***ClassNotFoundException*** :
Le driver n'est pas dans le classpath
 - **L'URL** n'est pas correcte.
Utiliser l'assistant d'*iReport* pour définir l'URL ou se référer au site de l'éditeur
 - Les **paramètres de connexion** ne sont pas corrects.
Vérifier le nom de la base, l'utilisateur et le mot de passe.



Requêtes SQL définies dans le rapport

- *jasperStudio* fournit plusieurs outils pour travailler avec JDBC et SQL
 - Test de la connexion JDBC
 - Découverte automatique des champs possibles avec leur type
 - Possibilité d'enregistrer automatiquement les champs
 - Un assistant pour mettre au point les requêtes (*Query Designer*)
 - Un prévisualiseur des données renvoyées par la requête

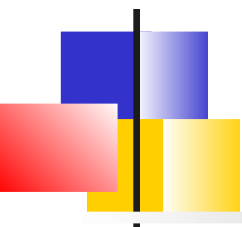
DataSet Dialog





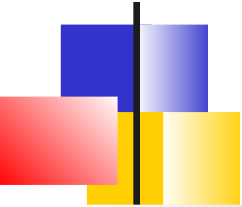
Tri et filtre

- Les enregistrements récupérés d'une source de données peuvent être triés et filtrés programmatiquement par JasperReport
 - L'expression du filtre doit retourner un booléen
 - Le tri est basé sur un ou plusieurs champs en pouvant indiquer pour chaque la direction du tri
- Lorsque l'on utilise des bases de données relationnelles, il est en général plus performant d'utiliser les clauses SQL appropriées



Champs

Champs

- 
- Les champs représentent l'unique façon de faire correspondre les données de la source avec les données du rapport.
 - Les champs doivent être déclarés dans le rapport via l'élément **<field>** en indiquant
 - L'attribut **name** : le nom de la colonne récupérée par la requêtes. Il doit être unique dans le rapport. Si plusieurs champs de la base ont le même nom, il faut utiliser des alias dans la requête SQL.
 - L'attribut **class** : Indiquant la classe de l'objet pouvant être utilisé dans le rapport.
 - Les éléments **<field>** sont placés sous l'élément **<jasperReport>**



Correspondance de type

Type SQL	Objet Java	Type SQL	Objet Java
CHAR	String	REAL	Float
VARCHAR	String	FLOAT	Double
LONGVAR CHAR	String	DOUBLE	Double
NUMERIC	java.math.BigDecimal	BINARY	byte[]
DECIMAL	java.math.BigDecimal	VARBINARY	byte[]
BIT	Boolean	LONGVARBINARY	byte[]
TINYINT	Integer	DATE	java.sql.Date



Exemple

```
<jasperReport>
```

```
  <field name="id" class="java.lang.Integer"/>
```

```
  <field name="name" class="java.lang.String"/>
```

```
  <field name="city" class="java.lang.String">
```

```
    <fieldDescription>
```

```
      Ville de naissance
```

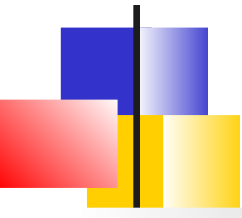
```
    </fieldDescription>
```

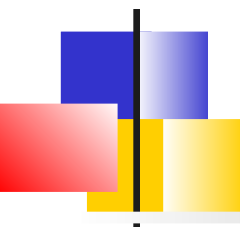
```
  </field>
```

```
....
```

```
</japserReport>
```

Règles sur les champs

- 
- Chaque champ doit correspondre alors à une colonne du *ResultSet*.
C'est à dire avoir le **même nom et un type compatible** avec la colonne de la base.
 - Si le nom indiqué ne correspond pas à une colonne du *ResultSet*, une exception sera lancée au moment de l'exécution
 - Les colonnes présentes dans le *ResultSet* mais n'ayant pas de champs associés ne seront pas accessibles dans le rapport.



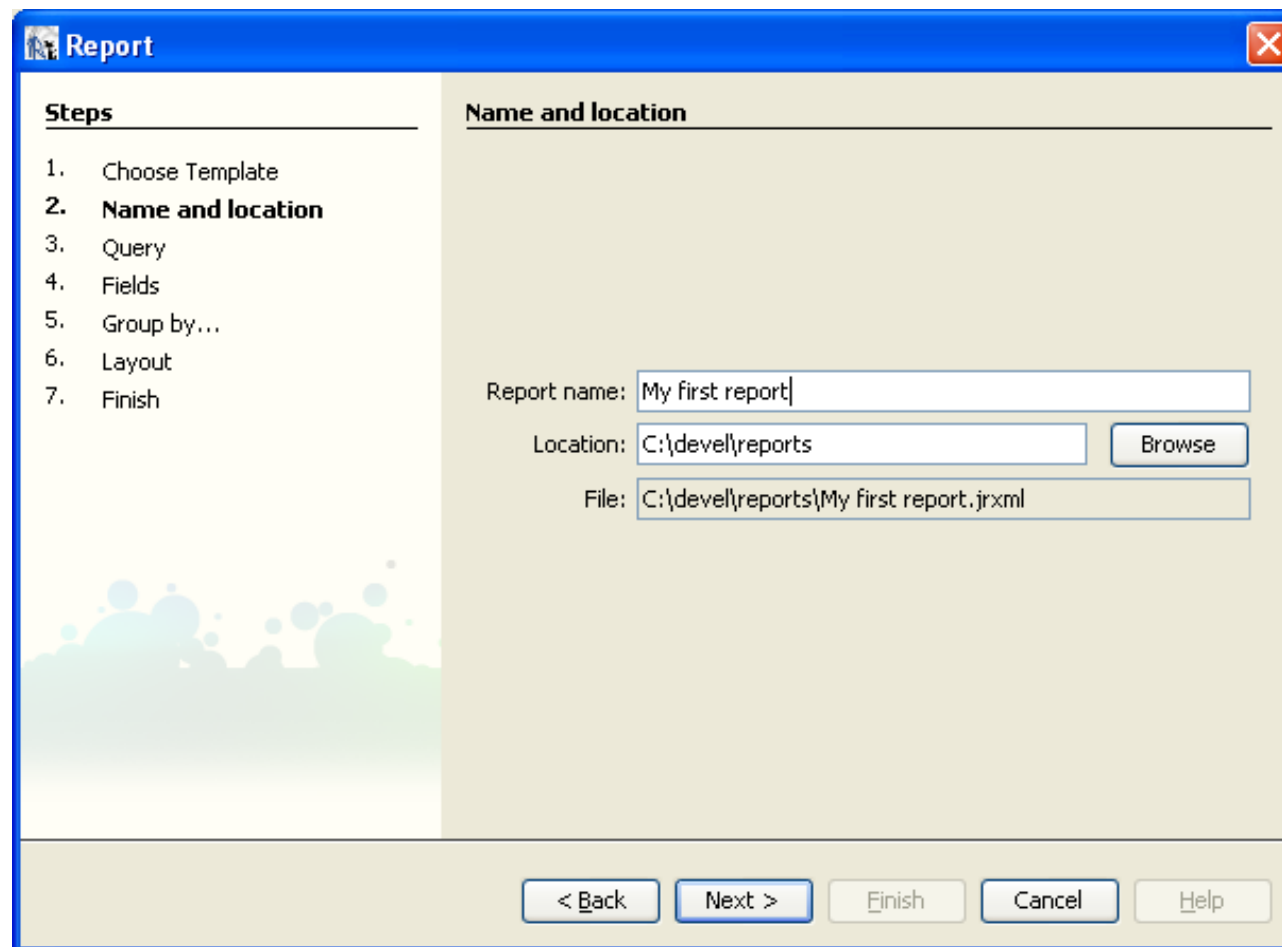
Premier rapport avec l'assistant de *JasperSoft Studio*



Création de rapport

- *jasperStudio* propose un assistant pour créer un rapport. L'assistant est basé sur des **gabarits** (*template*) ; il est possible de définir ses propres gabarits
- L'assistant s'effectue en 5 étapes :
 - Sélection d'un modèle
 - Nom et emplacement du rapport
 - Source de données et requête SQL
 - Champs utilisés
 - Les groupes éventuels

Etape 2 : Fichier



The image shows a 'Report' wizard window with a blue title bar and a close button. On the left, a 'Steps' list shows seven steps: 1. Choose Template, 2. Name and location (highlighted), 3. Query, 4. Fields, 5. Group by..., 6. Layout, and 7. Finish. The main area is titled 'Name and location' and contains three input fields: 'Report name' with the text 'My first report', 'Location' with 'C:\devel\reports' and a 'Browse' button, and 'File' with 'C:\devel\reports\My first report.jrxml'. At the bottom are five buttons: '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'.

Report

Steps

1. Choose Template
- 2. Name and location**
3. Query
4. Fields
5. Group by...
6. Layout
7. Finish

Name and location

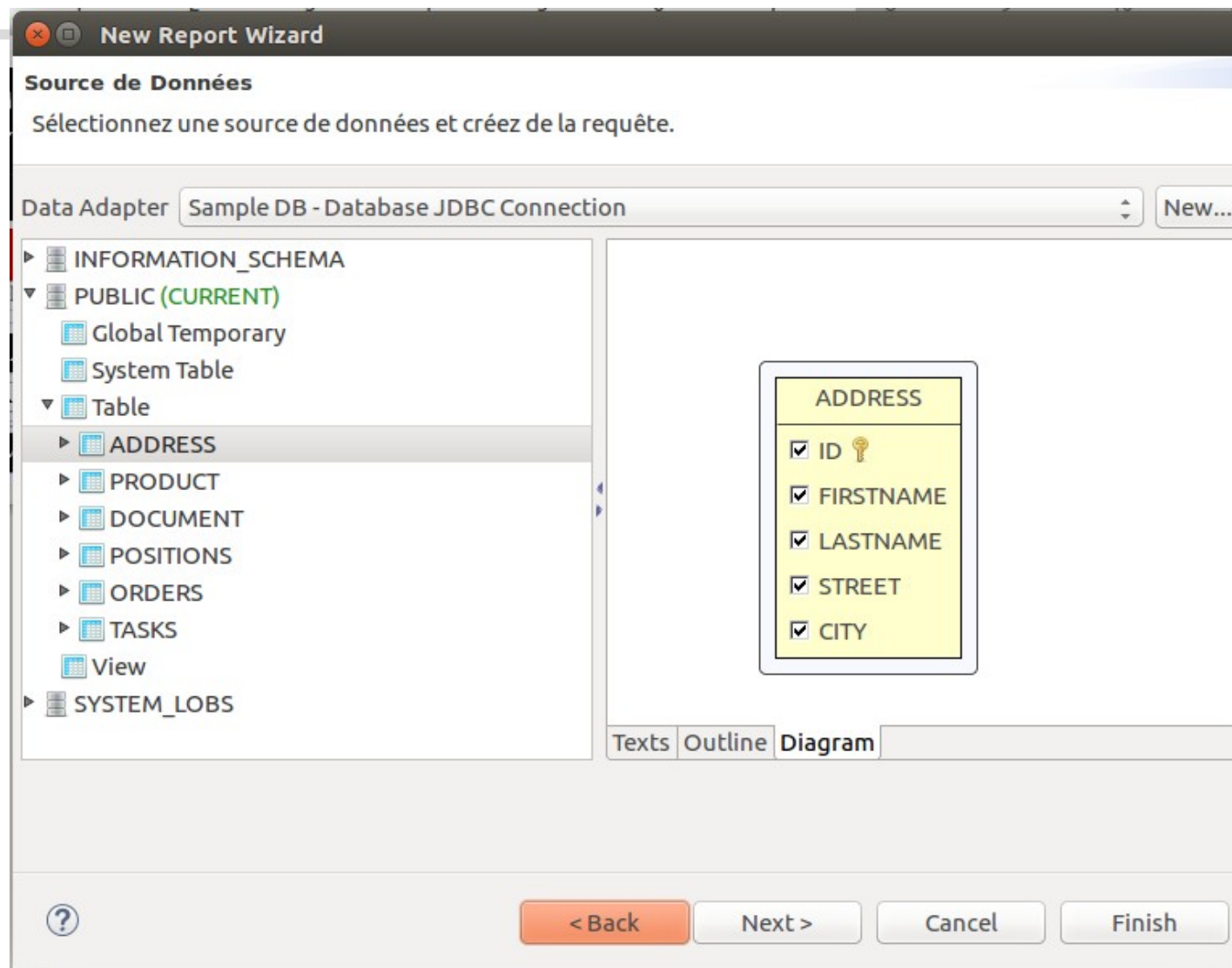
Report name:

Location:

File:

< Back Next > Finish Cancel Help

Etape 3 : Requête





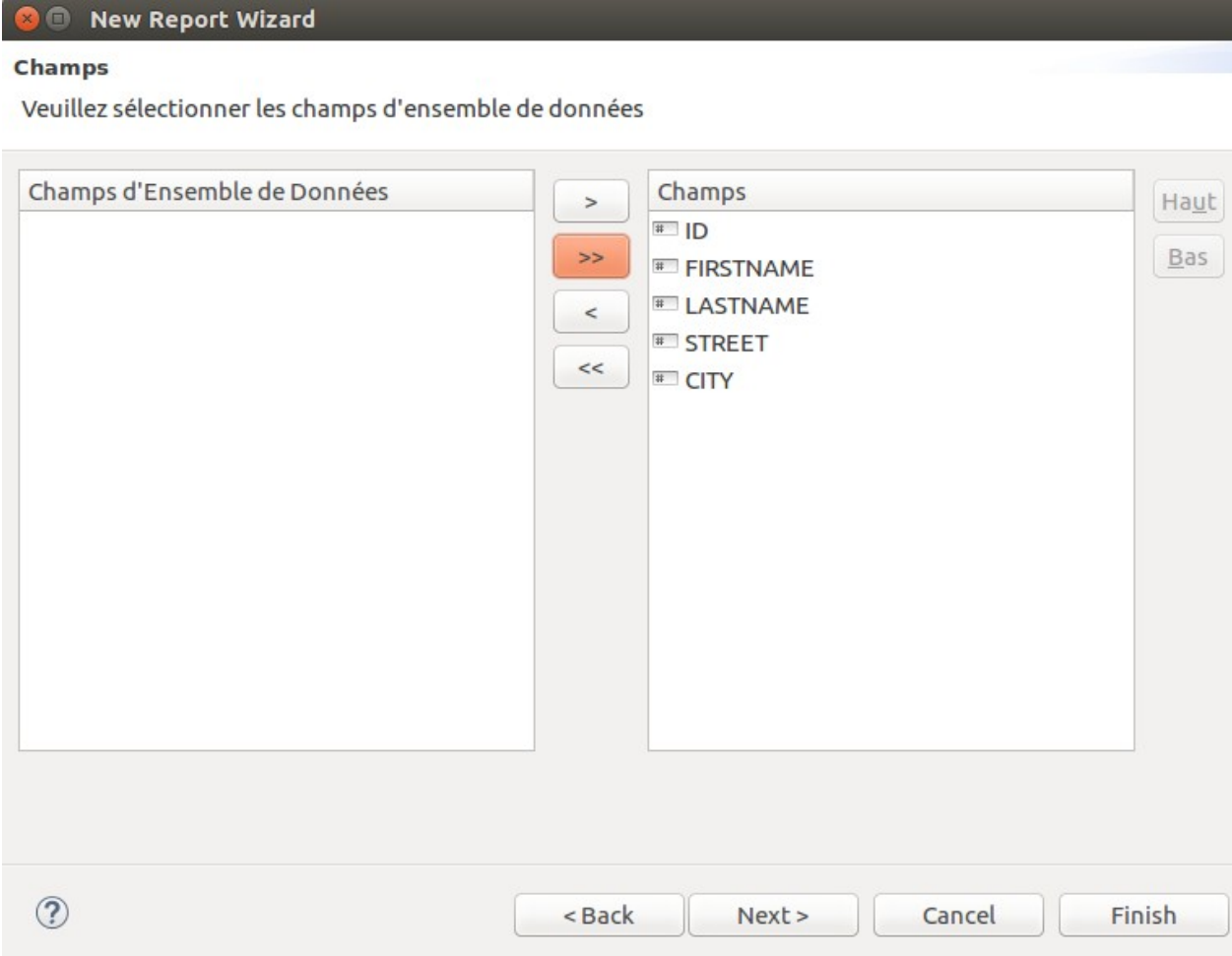

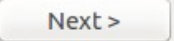


Etape 4 : Champs

New Report Wizard

Champs

Veuillez sélectionner les champs d'ensemble de données

Champs d'Ensemble de Données		Champs	
	>	ID	Haut
	>>	FIRSTNAME	Bas
	<	LASTNAME	
	<<	STREET	
		CITY	

Etape 5 : Groupe

New Report Wizard

Grouper Par
Veuillez sélectionner les champs à grouper par.

Champs d'Ensemble de Données

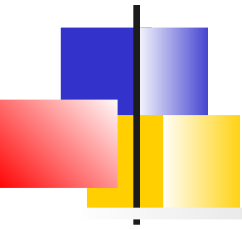
- ID
- FIRSTNAME
- LASTNAME
- STREET

Champs

- CITY

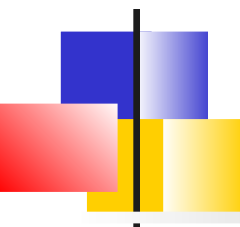
☐ Use the group fields as sort fields. Select this option if you want to aggregate all the group fields with the same value and not only the consecutive ones

Navigation: < Back, Next >, Cancel, Finish

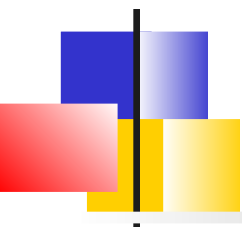


Composition de rapport

Dimension et bandes
Éléments simples
Paramètres, expressions, variables

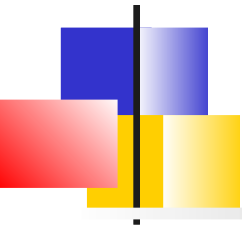


Dimension et Bandes



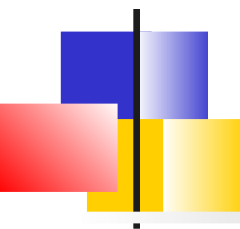
Unité de mesure et attribut

- Les premières propriétés d'un rapport sont ses dimensions
- L'unité par défaut est le pixel et peut être changée
- Les attributs concernent la dimension de la zone de dessin :
 - Largeur et hauteur de page.
 - Largeur et hauteur des différentes marges



Bandes

- Un rapport est divisé en **bandes horizontales**.
 - La construction d'un rapport consiste à définir le contenu et les dimensions de ses bandes.
 - Chaque bande contient des éléments de rapport comme des lignes, des rectangles, des images ou des champs texte.
 - La structure d'un document Jasper est prédéfini avec un ensemble de bandes de départ
 - Les bandes peuvent être ajoutées/supprimées du rapport
 - Une bande correspond section à l'élément `<band>` dans le fichier `.jrxml`



Structure d'un rapport

- La structure d'un rapport est basée sur les bandes suivantes :
 - **<background>** : Fond de page appliqué sur toutes les pages
 - **<title>** : Titre apparaissant une seule fois au début
 - **<pageHeader>**, **<pageFooter>** : Entête et pied de page
 - **<columnHeader>**, **<columnFooter>** : Pour les rapports multi-colonnes, entête et pied des détails de colonne
 - **<groupHeader>**, **<groupFooter>** : Entête et pied d'un groupe (0 ou plusieurs)
 - **<detail>** : Zone répétée pour chaque donnée. Possibilité de définir plusieurs sections *<detail>*
 - **<lastPageFooter>** : Pied de page pour la dernière page
 - **<summary>** : Affiché à la fin du rapport
 - **<noData>** : Texte alternatif lorsqu'il n'y a pas de données



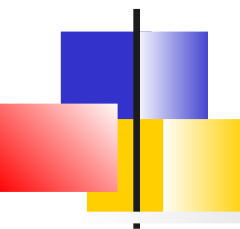
Bandes par défaut

	title	
	pageHeader	
	columnHeader	
	detail	
	columnFooter	
	pageFooter	
	summary	



Attributs d'une bande

- Attribut ***height***
 - C'est la hauteur de la bande dans la vue design. Cette hauteur peut varier lors de la fusion de données (si les données dépassent la hauteur spécifiée). C'est donc la hauteur minimale de la bande à la génération.
 - Lorsque cette valeur est modifiée, *iReport* vérifie si la valeur est acceptable en fonction des éléments que la bande contient.
- Attribut ***printWhenExpression***
 - Expression booléenne permettant de cacher la bande
 - L'expression est évaluée à chaque fois que la bande est utilisée lors de la génération. (1 fois pour le titre, à chaque page pour l'entête de page, à chaque enregistrement pour le détail).
- Attribut ***splitType*** (anciennement ***isSplitAllowed***)
 - ***stretch***: La bande est divisée sur différentes pages seulement si son contenu dépasse la hauteur précisée.
 - ***prevent***: La bande n'est pas divisée. (Seulement si sa taille dépasse la taille d'une page)
 - ***immediate***: La bande peut être divisée.



Colonnes

- Les bandes pilotées par les enregistrements de la source de données (détail, groupe, sous-rapport, ...) peuvent être organisées en colonnes afin d'optimiser l'espace disponible.
- L'attribut ***columnCount*** du rapport permet donc de fixer le nombre de colonnes :
 - Le remplissage des colonnes peut alors être vertical ou horizontal (Orientation)
 - La somme des marges, des largeurs de colonnes (*columnWidth*) et de chaque espace entre les colonnes doit être inférieure à la largeur de la page, sinon des erreurs de compilation peuvent apparaître.
 - Lors du design, les éléments (champs, images, etc...) sont placés uniquement dans la première colonne.

Design

Listing des employés

Listing des employés

Nom	Prénom
<code>\$F{FIRSTNAME}</code>	<code>\$F{FIRSTNAME}</code>
Nom	Prénom

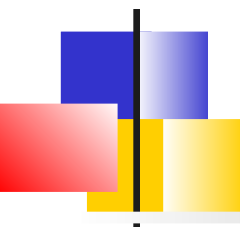
`NOW()`

Pied de Page

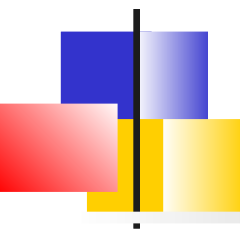


List of names on a two columns report

First name	Last name	First name	Last name
Laura	Steel	Sylvia	Fuller
Susanne	King	Susanne	Heiniger
Anne	Miller	Janet	Schneider
Michael	Clancy	Julia	Clancy
Sylvia	Ringer	Bill	Ott
Laura	Miller	Julia	Heiniger
Laura	White	James	Sommer
James	Peterson	Sylvia	Steel
Andrew	Miller	James	Clancy
James	Schneider	Bob	Sommer
Anne	Fuller	Susanne	White
Julia	White	Andrew	Smith
George	Ott	Bill	Sommer
Laura	Ringer	Bob	Ringer
Bill	Karsen	Michael	Ott
Bill	Clancy	Mary	King
John	Fuller	Julia	May
Laura	Ott	George	Karsen



Éléments simples



Introduction

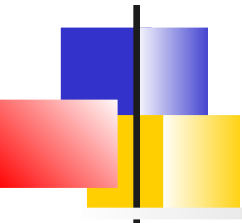
- Les éléments d'un rapport sont des objets d'affichage (Texte, Image, Forme, Graphique, ...) qui peuvent être dimensionnés, alignés, formatés etc.
- Tout contenu est créé via ces éléments.
- La palette de *jasperStudio* proposent un certain nombres d'éléments :
 - Ligne, Rectangle, Ellipse, Texte statique, Champ texte, Image, Cadre, Sous-rapport, Tableau, Liste, Tableau croisé, Graphique, Saut de page ou colonne
- Finalement, il est également possible d'étendre la bibliothèque d'éléments avec du code java

Hiérarchie des éléments

Les éléments sont organisés hiérarchiquement ; ils ont tous des propriétés communes et chaque niveau de la hiérarchie apporte ses spécificités

Timeseries Chart - Propriétés	
▼ Propriétés	
Left	0
Top	453
Width	555
Height	136
Forecolor	■ [0,0,0]
Backcolor	□ [255,255,255]
Opaque	<input type="checkbox"/>
Style	
Key	
Position Type	Fix Relative to Top
Stretch Type	No stretch
Print Repeated Values	<input checked="" type="checkbox"/>
Remove Line When Blank	<input type="checkbox"/>
Print In First Whole Band	<input type="checkbox"/>
Print When Detail Overflows	<input type="checkbox"/>
Print When Group Changes	
Print When Expression	
Properties expressions	No properties set
► Common chart properties	
▼ TimeSeriesPlot properties	
Show Lines	<input checked="" type="checkbox"/>
Show Shapes	<input checked="" type="checkbox"/>
Time Axis Label Expression	

Graphique: Time Series

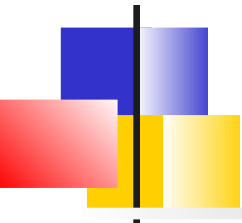


Positionnement et taille

- Les éléments sont positionnés via leurs attributs **top** et **left**
 - Ces valeurs sont relatives au container parent (bande ou frame)

=> Il est impossible de positionner un élément à cheval sur deux bandes

- La taille de l'élément est spécifié par les attributs **width** et **height**



Layout

Les éléments pouvant contenir d'autres éléments comme les bandes, les cadres, les cellules de table ou de tableau croisé peuvent être associés à des ***layouts***.

Les layouts ont des impacts sur la taille et le positionnement des éléments à l'intérieur du conteneur.

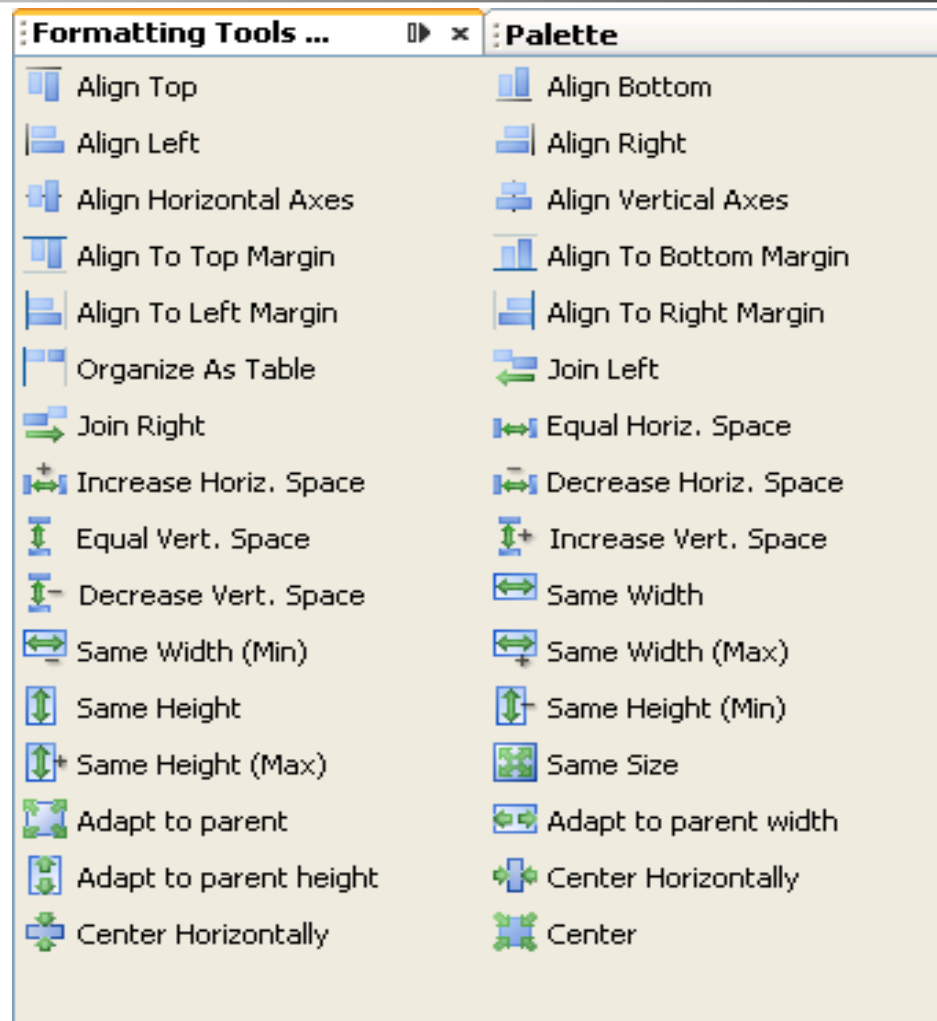


Types de layouts

JasperStudio propose 4 types de layout :

- **Free layout (défaut)** : Le développeur choisit librement la taille et la position des sous-éléments
- **Horizontal layout** : Les éléments sont disposés horizontalement dans le conteneur. Ils ont tous la même largeur
- **Vertical layout** : Les éléments sont disposés verticalement dans le conteneur. Ils ont tous la même hauteur
- **Grid layout** : Le conteneur spécifie un nombre de lignes et de colonnes et chaque élément indique sa position vis à vis de cette grille. Les cellules de la grille peuvent être fusionnées

Outils de formatage





Outils de formatage (2)

Outil	Description	Multi
Align (left, right, top, bottom)	Aligne en fonction des côtés du 1er élément	X
Align Axis (horizontal, vertical)	Aligne en fonction des axes du 1er élément	X
Align band (top, bottom)	Aligne en fonction de la bande parente	
Same (width, height, size)	Fixe les dimensions par rapport au 1er élément	x
Same max (width,height)	Fixe les dimensions par rapport à l'élément le plus grand	x
Same min (width,height)	Fixe les dimensions par rapport à l'élément le plus petit	x
Center (horizontally, vertically)	Centre par rapport au container	



Outils de formatage (3)

Outil	Description	Multi
Center in background	Centre par rapport à la page et positionne dans le fond	
Join sides (right, left)	Juxtapose horizontalement les éléments	x
Horizontal space	Modifie les espaces horizontaux entre éléments	x
Vertical space	Modifie les espaces verticaux entre éléments	x
Adapt to parent	Redimensionne en fonction du parent	
Organize as table	Espace les éléments de façon égale	x



Axe z

- L'ordre des éléments d'une bande est important car il définit l'ordre d'impression. C'est le **z-index**.
 - Si les positions d'un élément se superposent avec celles d'un de ces prédécesseurs, il risque de le recouvrir lors de l'impression.
 - Dans d'autres situations, on peut prévoir deux éléments à la même position mais un seul est affiché grâce à l'attribut *printWhenExpression*.
 - Certains exporters, comme l'exporter HTML ne supportent pas les éléments superposés et donc les ignore.



Attributs des éléments

- **key** : Nom unique de l'élément
- **x, y, width, height** : Positions
- **foreground, background, opaque** : Couleurs
- **style** : Style de l'élément.
- **isRemoveLineWhenBlank** : Suppression de l'espace si l'élément est vide
- **properties** : Propriétés arbitraires de l'élément.



Type de position

- L'attribut ***PositionType*** détermine comment les coordonnées sont modifiées lorsque la hauteur de la bande change (lors de la fusion de données).
- 3 valeurs sont possibles :
 - ***FixRelativeToTop*** : Valeur par défaut, les coordonnées ne changent jamais.
 - ***Float*** : L'élément est progressivement déplacé vers le bas par les éléments précédents qui augmente leur taille
 - ***FixRelativeToBot*** : La distance entre l'élément et le bas de la bande reste constant.



StretchType

- L'attribut ***StretchType*** définit comment calculer la hauteur de l'élément durant l'impression.
- 3 valeurs sont possibles :
 - ***NoStretch*** : Valeur par défaut. La hauteur ne doit pas changer
 - ***RelativeToBandHeight*** : La hauteur de l'élément est augmentée proportionnellement à l'augmentation de la hauteur de la bande. (Utile pour les lignes verticales simulant des bordures de tableau)
 - ***RelativeToTallestObject*** : La hauteur de l'élément est modifiée en fonction de la déformation du plus grand élément.



Impression

- ***isPrintRepeatedValues*** : Impression si valeur égale au précédent enregistrement
- ***isPrintInFirstWholeBand*** : Garantit l'impression de l'élément lors d'une nouvelle page ou nouvelle colonne.
- ***isPrintWhenDetailOverflows*** : Imprimé sur la page suivante, lorsque la bande ne tient pas sur la page courante
- ***printWhenGroupChanges*** : Imprimé lors d'une rupture de groupe
- ***printWhenExpression*** : Expression booléenne conditionnant l'impression

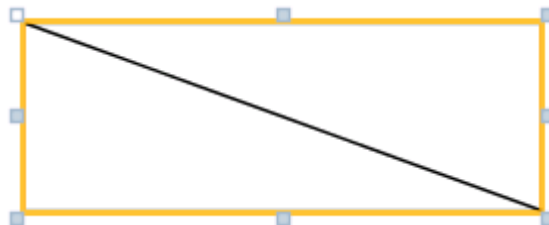


Éléments graphiques

- Les éléments graphiques sont :
 - Les ligne
 - Les rectangles
 - Les ellipses
- Ces éléments ont des attributs supplémentaires communs
 - **pen** : Le trait dessinant la forme
 - **fill** : Le remplissage de la forme

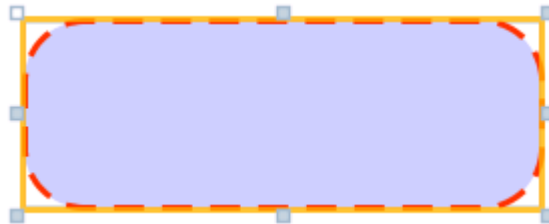
Ligne

- Une ligne est définie par la diagonale d'un rectangle
- L'attribut ***direction*** détermine la diagonale:
 - *TopDown*
 - *BottomUp*



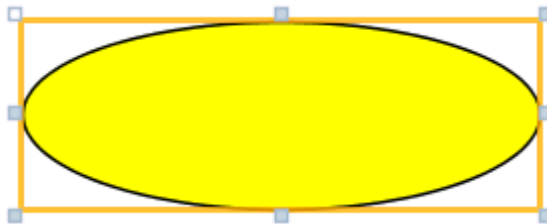
Rectangle / Rectangle arrondi

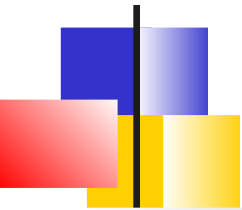
- Possibilité d'avoir des coins arrondis en utilisant l'attribut ***radius***



Ellipse

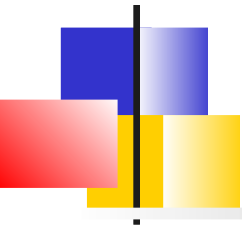
- Défini également à partir d'un rectangle, cet élément n'a pas d'attributs spécifiques





Cadre

- Un **cadre** (*frame*) est un élément qui contient d'autres éléments. Éventuellement, il peut définir une bordure
- On peut imbriquer autant de niveau que l'on veut.
- Par rapport à l'utilisation d'un rectangle, un cadre permet de :
 - Déplacer tous les éléments du cadre en même temps
 - Appliquer un layout particulier
 - Appliquer un style (il sera appliqué sur tous les éléments contenus dans le cadre)
 - Pas besoin de jouer sur l'axe Z



Images

- C'est l'objet le plus complexe des objets graphiques. Il sert à :
 - Insérer un fichier image binaire (JPEG, PNG, GIF)
 - Ou définir une zone de *Canvas*.
Un code Java pourra alors dessiner le graphique ou produire une image binaire.



ImageExpression

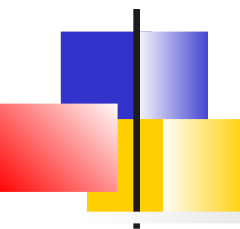
- Les images ne sont pas stockées avec le rapport, seule sa référence via l'élément **<imageExpression>** y est présente.
- L'attribut **class** de la balise indique le type de l'expression ; les principales valeurs sont :
 - **java.lang.String** : Emplacement et nom du fichier.
Ex : */home/public/images/logo-footer.gif*
 - **java.io.File** : Objet Java File.
Ex : *new java.io.File("c:\\myImage.jpg")*
 - **java.net.URL** : L'URL de l'objet
Ex : *new java.net.URL("http://127.0.0.1/test.jpg")*



Portabilité

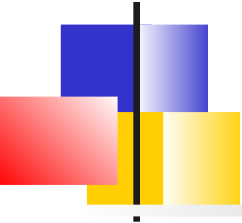
- Lorsque l'on définit une image en précisant son emplacement, il faut que cet emplacement soit valide dans l'environnement de développement **ET** de production
- Les meilleures pratiques pour éviter les problèmes en production sont :
 - Utiliser un **paramètre** dans l'expression indiquant le répertoire où sont placées les images :
`$P{MY_IMAGES_DIRECTORY} + "myImage.png"`
Au moment de la génération, l'application métier ou JasperServer positionne ce paramètre.
 - Utiliser le **classpath** JAVA. (Recommandé)
Dans ce cas, l'expression de l'emplacement est relative au classpath et le déploiement de l'application inclut les images dans l'archive de l'application.

Ces méthodes sont utilisées pour les autres types de ressources externes (sous-rapport, police externe, scriptlet, ressource bundle, ...)



Attributs de chargement

- ***isLazy*** : L'image n'est chargée qu'à l'exécution du rapport
- ***isUsingCache*** : L'image est gardée en mémoire pour être réutilisée



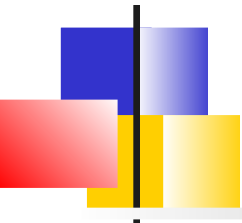
Comportement en cas d'erreur

- ***onErrorType*** : Spécifie le comportement si l'image ne peut pas être chargée :
 - ***Error*** : Une exception Java est lancée.
 - ***Blank*** : L'exception est ignorée et l'image est remplacée par un rectangle blanc.
 - ***Icon*** : L'exception est ignorée et un icône de remplacement est affiché.



Autres attributs

- **Scale** : Définit comment l'image se comporte par rapport aux dimensions de l'élément :
 - **Clip** : L'image n'est pas modifiée, et seule la partie correspondant aux dimensions de l'élément est affichée
 - **FillFrame** : L'image remplit le cadre
 - **RetainShape** : L'image est adaptée en conservant les proportions
 - **RealHeight** : Conserve la hauteur réelle de l'image (*JRRenderable*)
 - **RealSize** : Conserve la taille réelle de l'image (*JRRenderable*)
- Alignements verticaux et horizontaux (**halign** et **valign**)



Textes

- Deux éléments sont prévus pour afficher du texte :
 - Les **textes statiques** dont la valeur ne change pas et est connue au moment du design :
<staticText>
 - Les **champs textes** dont la valeur est fournie sous forme d'expression :
<textField>



Attributs

- Police : *jasperStudio* présente toutes les polices trouvées dans le système mais attention à la compatibilité au moment de la génération.
- Taille de police (entier), Gras et italique, Souligné, Barré
- Alignements horizontaux et verticaux
- Rotation : None, Left , Right, UpsideDown
- Espacement des lignes : Simple, 1.5 ou Double
- Markup : Utilisation d'un langage de balise (HTML ou RTF)



Types d'expressions valides

- Le type retourné par l'expression est indiqué dans l'attribut *class* de l'élément *textFieldExpression*
- Les types valides sont :
 - ***java.lang.String*** : Chaînes de caractères
 - ***java.util.Date, java.sql.Timestamp, java.sql.Time*** : Dates et heures
 - ***java.lang.Boolean*** : Booléen
 - ***java.lang.Long, java.lang.Double, java.lang.Short, , java.lang.Float, java.math.BigDecimal, , java.lang.Integer*** : Chiffres
 - ***java.lang.Byte*** : Octet
 - ***java.io.InputStream*** : Flux d'octets
 - ***java.lang.Object*** : Objet Java



Attributs d'un *TextField*

- ***isBlankWhenNull*** : Si l'expression est *null*, la chaîne vide est utilisée (à la place de « *null* »)
- ***evaluationTime*, *evaluationGroup*** : Détermine dans quelle phase l'expression est évaluée et éventuellement pour quel groupe
- ***isStretchWithOverflow*** : Le champ texte s'adapte au contenu en cas de dépassement
- ***pattern*** : Le pattern à appliquer pour formater la valeur de l'expression. Dépend du type
 - Par exemple pour une date :
"dd/MM/yyyy 'à' HH:mm:ss z"
04/07/2001 à 12:08:56 PDT



Polices

Java et donc JasperReport distingue 2 types de Police :

- Les polices physiques installées sur le système
- Les polices logiques : *Serif*, *SansSerif*, *Monospaced*, *Dialog* et *DialogInput*. Elles n'ont pas de fichier installé sur le système mais elles sont associés à une des polices physique. La correspondance peut dépendre du système !!



Bonne pratique

L'utilisation de police système comporte toujours le risque qu'une police utilisée lors du développement ne soit pas présente dans l'environnement de production.

La bonne pratique est d'embarquer les fichiers de polices dans l'archive déployée.

Cette technique appelé « **Font Extension** » nécessite de fournir un fichier jar contenant les fichiers polices, un fichier *properties* et un descripteur XML associant les fichiers polices aux locales et style de police (italique, gras, ...)

Le jar peut être généré directement à partir de *iReport* après avoir ajouté la police voulue

Voir l'exemple dans */demo/samples*



Ajout de Police

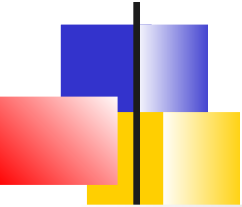
- Pour utiliser une police TrueType spécifique, il faut l'installer :
Windows → Preferences → Font
- Bien sûr, la police doit être présente au design **et** à la génération
- Il est également possible de saisir directement le nom d'une police qui ne sera présente qu'au moment de la génération







Liens Hypertexte

- Les éléments images, textes et graphiques peuvent être utilisés comme des ancres ou comme des liens hypertextes vers des URLs externes ou ancres internes
- Seuls les formats HTML et PDF permettent de créer des liens Hypertexte
- La fenêtre des dialogue « Hyperlink » permet de préciser :
 - L'expression de l'ancre
 - Un niveau de signet (utile pour l'exportation PDF)
 - La cible du lien
 - Le type de lien
 - Une expression pour le tooltip




Propriété Hyperlink





 **Image: Image** Search Property ▾

 Apparence  Bordures  Image »

☐ **Anchor and Bookmark**




Anchor Name Expression   




Bookmark Level  




☐ **Hyperlink**

Cible du Lien ▾

Type du Lien ▾

Expression Référence d'Hyperlien   

Hyperlink When Expression   

Expression Info-Bulle d'Hyperlien   



Type des hyperliens

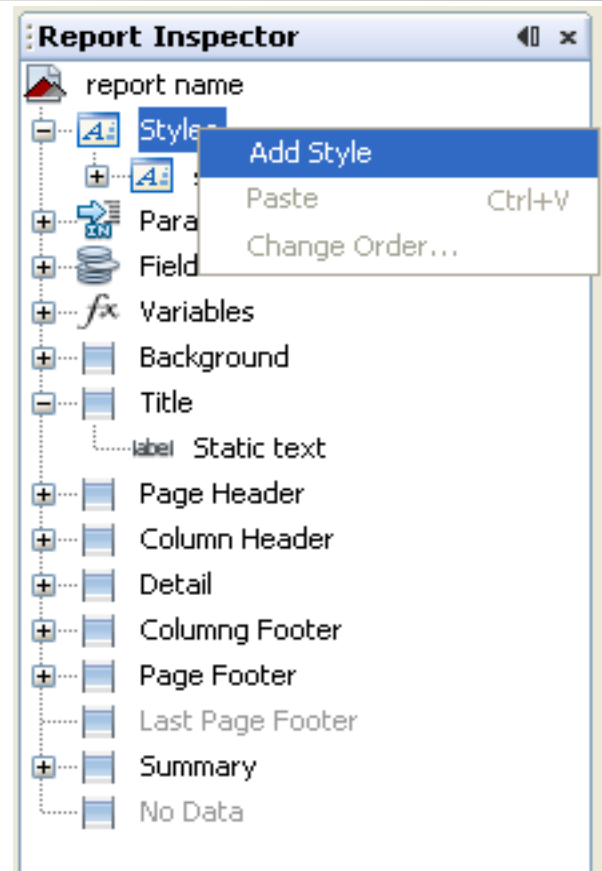
- JasperReport permet de définir différents types de liens :
 - **Reference** : Indique une URL externe
 - **LocalAnchor** : Lien vers une ancre interne
 - **LocalPage** : Lien vers un n° de page
 - **RemoteAnchor** : Lien vers une ancre externe
 - **RemotePage** : Lien vers une n° de page d'un autre rapport
 - **ReportExecution** : Utilisé pour lier vers un rapport déployé sur JasperServer



Styles

- Les styles sont utilisés pour regrouper un ensemble de propriétés de présentation et peuvent ensuite être appliqués aux éléments du rapport
- => Ainsi, la modification d'un style peut affecter un ensemble d'éléments du rapport
- Les éléments peuvent individuellement surcharger les propriétés définies par leur style
- Il est possible de définir un style par défaut au niveau du rapport.

Ajout de style

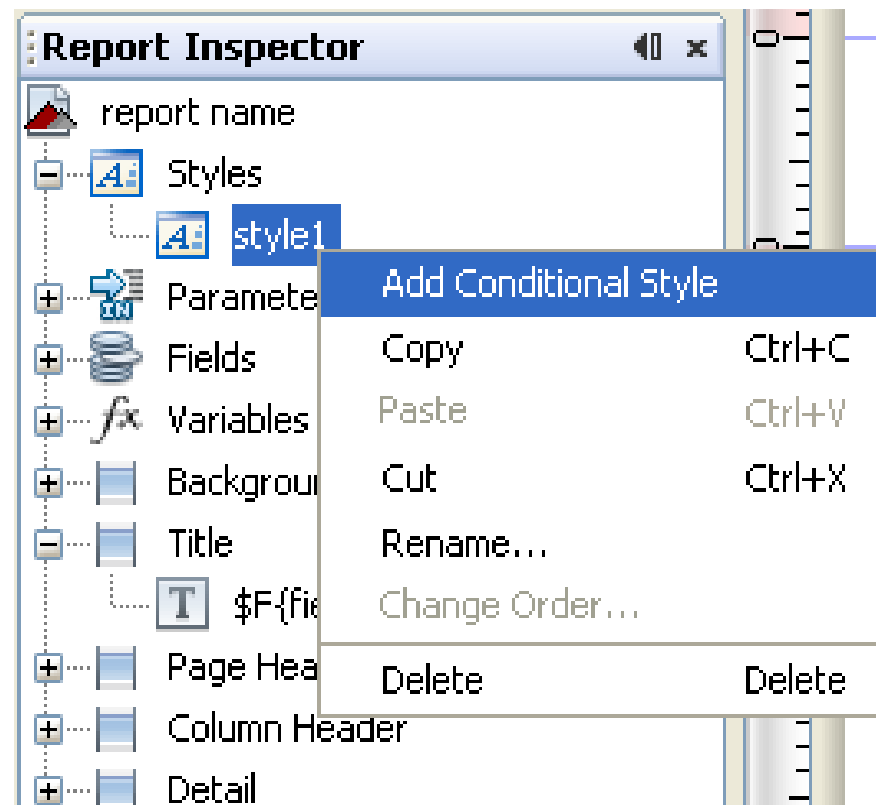




Styles conditionnels

- Il est possible de créer des rapports où le style appliqué change dynamiquement en fonction de variable ou d'expression du rapport
Exemple : Mettre en rouge un texte en fonction d'une condition
- *jasperStudio* permet d'associer à un style, un **style conditionnel** qui lorsque certaines conditions seront remplies surchargera certaines propriétés du style
- La condition peut s'exprimer en fonction de toutes les propriétés du rapport (paramètres, champs, variables, ...)

Ajout de style conditionnel



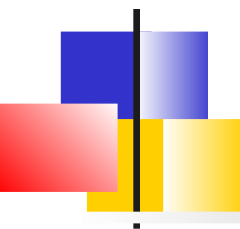


Librairie de styles

Au lieu de définir les styles
indépendamment dans chaque rapport.

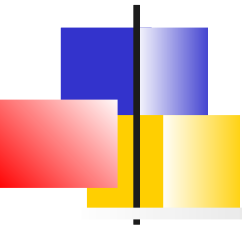
Il est possible de référencer une librairie
de style dans un rapport

La librairie de styles (**Style Template
File**) est un fichier **.jrtx** complètement
autonome et ayant un éditeur
spécifique



Notions de paramètres, d'expressions, de variables

Paramètres



- Les **paramètres** sont des données passées lors de la génération du document dynamique qui ne proviennent pas de la source de données:
 - Par exemple des données de présentation comme le nom de l'utilisateur, le titre du rapport, ...
 - Des paramètres de la requête SQL. Les paramètres agissent alors comme des filtres sur les données récupérées.
Exemple : une période, un montant, etc ...
 - Des paramètres servant à l'évaluation d'expression

Déclaration

- La déclaration d'un paramètre dans un design consiste à spécifier son **nom** et son **type** (classe Java):
- Dans le fichier *.jrxml*, cela donne :

```
<parameter name="ReportTitle" class="java.lang.String"/>
```

```
<parameter name="MaxOrderID" class="java.lang.Integer"/>
```

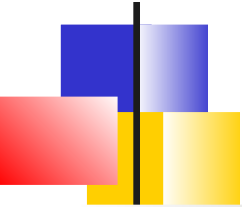
```
<parameter name="SummaryImage" class="java.awt.Image"/>
```



Attributs additionnels

- L'élément ***defaultValueExpression*** permet de spécifier une valeur par défaut.
Dans cette expression, il n'est pas possible d'utiliser des champs ou variables.
- Les attributs ***isForPrompting***, ***Description*** et ***properties*** ne sont pas utilisés directement par JasperReport mais peuvent être utilisés par d'autres applications externes :
 - pour déterminer si ce paramètre doit être fourni interactivement.
 - Une documentation
 - Stockée un ensemble de propriétés liées au paramètre. Par exemple, la description du champ en différents langages

Utilisation dans une requête

- 
- Les paramètres peuvent être utilisés dans les requêtes SQL :

- Comme paramètres normaux d'un *java.sql.PreparedStatement* :

*SELECT * FROM Orders WHERE CustomerID = **\$P{OrderCustomer}***

- Ou comme tout ou partie de la requête SQL

*SELECT * FROM **\$P!{TablesClause}***



Clause IN et NOT IN

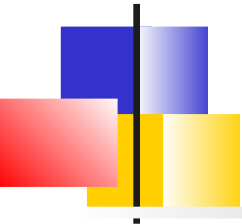
- *JasperReports* propose une syntaxe spécifique à utiliser dans les clauses IN et NOT IN des requêtes SQL.

Exemple :

```
SELECT * FROM ORDERS WHERE $X{IN,  
SHIPCOUNTRY,myCountries}
```

- La clause `$X{}` prend 3 paramètres :
 - Le type de la fonction à appliquer (IN ou NOT IN)
 - Le nom du champ à évaluer
 - Le nom du paramètre

Paramètres prédéfinis



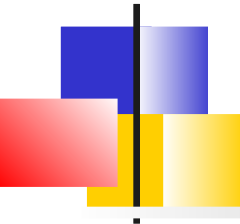
- JasperReports propose également des paramètres prédéfinis directement utilisable dans les rapports
 - **REPORT_PARAMETERS_MAP** : Une map contenant tous les paramètres prédéfinis.
 - **REPORT_CONNECTION** : Contient la connexion JDBC fournie par l'application.
 - **REPORT_DATA_SOURCE** : Contient l'instance de la source de données fournie par l'application
 - **REPORT_MAX_COUNT** : Un entier permettant de limiter la taille de la source de données
 - **REPORT_SCRIPTLET** : Contient l'instance d'un code Java éventuellement fournie par l'application



Paramètres prédéfinis

- **REPORT_LOCALE** : La locale (*java.util.Locale*) utilisée pour la génération
- **REPORT_RESOURCE_BUNDLE** : Le bundle contenant les labels localisés du rapport
- **REPORT_TIME_ZONE** : Le fuseau horaire (*java.util.TimeZone*) à utiliser pour le formatage des dates
- **REPORT_VIRTUALIZER** : L'instance de type *JRVirtualizer* utilisé pour la pagination
- **REPORT_CLASS_LOADER** : Le classloader à utiliser pendant la génération du rapport
- **IS_IGNORE_PAGINATION** : Flag. Si *true* le rapport est généré sans saut de page

Expressions



- Les **expressions** sont utilisées pour exprimer la valeur de certains éléments du rapport : variables, critère de regroupement, ou même tout simplement paramètre visuel du rapport
- Elles peuvent utiliser des opérateurs et les méthodes du langage
- Les langages disponibles sont Java, Javascript et Groovy
- Les expressions peuvent faire référence à des paramètres, des champs ou des variables

Spécification des expressions

- La notation **$\${}$** permet de faire référence à des champs, des variables, des paramètres ou des *resourceKey*
 - **$\$F\{<fieldName>\}$** pour un champ
 - **$\$V\{variableName\}$** pour une variable
 - **$\$P\{parameterName\}$** ou **$\$P!\{parameterName\}$** pour un paramètre.
 - **$\$R\{resourceKey\}$** pour la localisation



Types des expressions

- Le type de l'expression est souvent déterminé par son contexte d'utilisation (condition, calcul, texte, ...)
- Si le langage est Java, les types supportés sont : *Boolean*, *Byte*, *Integer*, *Long*, *Float*, *Double*, *String*, *Date* et *Object*
- En Javascript et Groovy, les expressions ne sont pas typées



Opérateurs

Opérateur	Description
+	Ajout de 2 nombres ou concaténation de 2 chaînes de caractères
-	Soustraction
*	Multiplication
/	Division
%	Modulo (Reste d'une division entière)
	Opérateur logique OU
&&	Opérateur logique ET
== et !=	Test d'égalité ou de différence (Attention en Java, différent de la méthode equals())



Expression conditionnelle

- Les expressions de type if-else peuvent être spécifiées en utilisant la syntaxe :
(<condition>) ? <value on true> : <value on false>
- Exemple :
(($\$F\{name\}.length()$ > 50) ? $\$F\{name\}.substring(0,50)$: $\$F\{name\}$)
- Cette syntaxe est réursive comme dans :
***(($\$F\{name\}.length()$ > 50) ?
(($\$F\{name\}.startsWith("A")$)) ? "AAAA" : "BBB") :
 $\$F\{name\}$)***
- Il est toujours possible de déléguer des expressions plus complexes à une méthode statique d'une classe Java
MyFormatter.toRomanNumber($\$F\{MyInteger\}.intValue()$)



Éléments des expressions

- Différents éléments XML peuvent définir des expressions :
 - **<variableExpression>** : Définir la valeur d'une variable. Se place à l'intérieur d'un élément *<variable>*
 - **<initialValueExpression>** : Définir la valeur initiale d'une variable. Se place à l'intérieur d'un élément *<variable>*
 - **<groupExpression>** : Critère de regroupement (ou de rupture) pour les groupes. Se place à l'intérieur d'un élément *<group>*.
 - **<printWhenExpression>** : Expression booléenne agissant sur l'impression. Se place à l'intérieur des éléments *<band>*, *<reportElement>*.
 - **<imageExpression>** : Expression déterminant l'image à insérer. Se place à l'intérieur d'un élément *<image>*
 - **<textFieldExpression>** : Expression déterminant une valeur texte. Se place à l'intérieur d'un élément *<textField>*



Évaluation des expressions

- Par défaut, les expressions sont évaluées immédiatement lorsque le moteur génère le rapport.
- On peut modifier ce comportement en modifiant l'attribut ***evaluationTime*** qui peut prendre les valeurs suivantes :
 - ***Now*** : Immédiatement (valeur par défaut)
 - ***Report*** : A la fin de la génération
 - ***Page*** : A la fin de la page
 - ***Column*** : A la fin de la colonne
 - ***Group*** : A la fin de la groupe
 - ***Band*** : A la fin de la bande dans laquelle se trouve l'expression
 - ***Auto*** : Lorsque l'expression dépend de variables interdépendantes, on laisse le choix au moteur pour choisir le bon moment pour l'évaluation

Variables

- Une **variable** est un objet construit à partir d'une expression qui peut être réutilisée à plusieurs endroits dans un rapport en faisant référence à son nom.

```
<variable name="FirstLetter" class="java.lang.String">  
  <variableExpression>  
    <![CDATA[$F{ShipCountry}.substring(0, 1).toUpperCase()]]>  
  </variableExpression>  
</variable>
```

- Dans son expression, une variable peut référencer une autre variable si cette dernière a été déclarée auparavant.



Calculs prédéfinis

- *JasperReport* permet d'effectuer certains types de calcul d'agrégation prédéfinis : *count*, *sum*, *average*, *lowest*, *highest*, *variance*, etc.
- Par exemple :

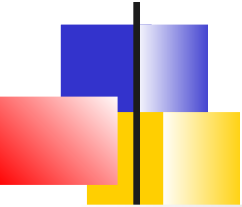
```
<variable name="QuantitySum"  
    class="java.lang.Double" calculation="Sum">  
    <variableExpression>$F{Quantity}</variableExpression>  
</variable>
```



Réinitialisation de variables

- Il est possible d'indiquer un niveau pour lequel la variable sera réinitialisée via l'attribut **resetType**. Les niveaux possibles sont :
 - **report** : Initialisé une fois au début du rapport (défaut)
 - **page** : Réinitialisé à chaque page
 - **column** : Réinitialisé à chaque colonne
 - **group** : Réinitialisé à chaque groupe.
- Par exemple, pour calculer le total d'un champ quantité pour chaque page

```
<variable name="QuantitySum" class="java.lang.Double"  
    resetType="Page" calculation="Sum">  
    <variableExpression>$F{Quantity}</variableExpression>  
    <initialValueExpression>new Double(0) </initialValueExpression>  
</variable>
```



Valeurs prenant part au calcul

- Par défaut, toutes les valeurs du *ResultSet* sont prises en compte pour le calcul de la variable
- Quelquefois, il est nécessaire de modifier ce comportement en jouant sur l'attribut ***IncrementType*** qui peut prendre les valeurs suivantes :
 - ***None*** : Pour chaque enregistrement
 - ***Page*** : Pour chaque page
 - ***Column*** : Pour chaque colonne
 - ***Group*** : Pour un chaque rupture d'un groupe particulier
 - ***Report*** : A la fin du rapport



Variables prédéfinies

- JasperReport fournit des variables prédéfinies :
 - **PAGE_NUMBER** : Le numéro de la page courante
 - **COLUMN_NUMBER** : Le numéro de la colonne courante
 - **REPORT_COUNT** : Nombre total d'enregistrements de la source de données
 - **PAGE_COUNT** : Nombre total d'enregistrements pour la génération de la page courante
 - **COLUMN_COUNT** : Nombre total d'enregistrements pour la génération de la colonne courante
 - **GroupName_COUNT** : Nombre total d'enregistrements pour la génération du groupe courant
- Voir la documentation de *JRVariable* pour une liste exhaustive



Propriétés

- Les **propriétés** sont des paires nom/valeur qui peuvent être associées au rapport ou aux éléments du rapport.
- Elles peuvent être utilisées à toutes sortes de propos : configurer un exporter, préciser un thème pour un graphique, crypter un PDF, ...
- JasperReport utilise déjà un certains nombre de propriétés dont la valeur par défaut peut être redéfinie.
 - Par exemple :
net.sf.jasperreports.text.truncate.at.char
définit comment un texte est coupé



Ressource Bundle

- Les **resource bundles** sont utilisés pour internationaliser un rapport, cela consiste en un ensemble de fichiers (1 par langue) contenant les traductions des textes apparaissant dans le rapport.
- Les fichiers respectent une norme de nommage :
 - Ils sont suffixés par le code pays et éventuellement langue (par exemple, `_fr_FR` pour Français/France)
 - Ils ont l'extension `.properties`
Exemple : `labels_fr_FR.properties` ou `labels_fr.properties`
- Le contenu du fichier est composé de lignes :
resourceKey= translated_label
 - **labels_fr.properties :**
`footer.confidential = Ces données sont confidentielles`
`generated.by = Généré par {0}`
 - **labels_en.properties :**
`footer.confidential = These data are confidential`
`generated.by = Generated by {0}`



Association avec le rapport

- L'association d'un *ResourceBundle* avec un rapport s'effectue :
 - Soit en utilisant l'attribut ***resourceBundle*** du rapport
 - Soit à l'exécution en fournissant une valeur pour le paramètre prédéfini **`$P{REPORT_RESOURCE_BUNDLE}`**.
- La langue courante quant à elle peut être surchargée par le paramètre **`$P{REPORT_LOCALE}`**.
- Dans le résultat généré, la librairie préserve l'information de direction du texte afin que les documents générés dans des langages comme l'Arabe ou l'hébreu puissent être correctement rendus.



Utilisation dans Jasper

- Pour localiser un label, on utilise :
 - la syntaxe **`$R{<bundleKey>}`** dans les expressions.
 - Ou la méthode **`msg()`** qui offre des fonctionnalités similaires à la classe *java.text.MessageFormat* en autorisant 3 paramètres de messages.

Exemples

- `$R{footer.confidential}`
- `msg($R{generated.by},$P{USER_LASTNAME})`