eBook Gratuit

APPRENEZ jasper-reports

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#jasper-reports

Table des matières

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: jasper-reports

It is an unofficial and free jasper-reports ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official jasper-reports.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Démarrer avec les rapports Jasper

Remarques

Il existe plusieurs bibliothèques utilisées par *l'API Java JasperReports* pour créer des rapports avec Java:

- **DynamicReports**
- **DynamicJasper**

Ces bibliothèques / frameworks peuvent générer des rapports "à la volée" avec ou sans utiliser le modèle de rapport (fichier *jrxml*)

Versions

Bibliothèque *JasperReports*

Version	Date de sortie
6.3.0	2016-06-20
6.2.0	2015-11-11
5.6.0	2014-05-27
5.5.0	2013-10-24
5.0.4	2013-03-26
5.0.0	2012-11-12
4.8.0	2012-11-05
4.7.0	2012-07-02
4.6.0	2012-05-21
4.5.0	2011-12-06
4.1.1	2011-04-18
4.0.0	2010-12-31
3.7.6	2010-10-27

Version	Date de sortie
3.7.5	2010-09-22
3.7.0	2009-12-08
3.6.0	2009-08-31
3.5.3	2009-07-29
3.5.0	2009-03-25
3.1.4	2009-02-10
3.1.2	2008-11-04
3.1.0	2008-09-17
3.0.1	2008-08-07
3.0.0	2008-05-19
2.0.5	2008-03-12
2.0.3	2007-12-12
2.0.0	2007-08-14
1.3.4	2007-06-11
1.3.0	2006-12-22
1.2.8	2006-11-14
1.2.0	2006-02-06
1.1.0	2005-10-21
1.0.3	2005-10-10
1.0.0	2005-07-20
0.6.8	2005-05-31
0.2.3	2002-02-06

IDE pour la conception de rapports

La version actuelle de designer est basée sur *Eclipse* : *Jaspersoft Studio* .

La version précédente de designer était basée sur *NetBeans* : *iReport Desigher* .

La première version de *iReport Designer* était une application indépendante - *iReport Classic*

Examples

Installation ou configuration

Bibliothèque JasperReports

JasperReports est un outil de reporting basé sur Java open source. La bibliothèque *JasperReports* peut être téléchargée depuis la communauté Jaspersoft pour la dernière version .

> Dans les versions récentes, les `pom.xml` tiers dans le dossier lib **ne** sont **pas** distribués, ils doivent être téléchargés à partir de référentiels publics, voir `pom.xml` distribué pour les dépendances. Maven peut être utilisé pour récupérer toutes les dépendances, y compris celles qui sont transitoires dans le dossier cible / dépendance.

```
mvn dependency:copy-dependencies
```

Jaspersoft Studio (IDE)

Jaspersoft Studio est le client de conception officiel de JasperReports - construit sur la plateforme Eclipse - pour remplacer iReport Designer.

iReport Designer (IDE)

iReport Designer est le concepteur de rapports précédent pour JasperReports. La version 5.6.0 (sortie en mai 2014) était la dernière version officielle; le support fournisseur a pris fin fin 2015.

Ressources de JasperReport Commuity

FAQ de la bibliothèque JasperReports

- FAQ

Code source

- Code source de la bibliothèque JasperReports

Des tutoriels

- Point de tutoriels
- JasperReports Ultimate Guide

Des échantillons

- Référence de l'échantillon

Les références

- Documentation officielle
- Wiki de la communauté

Bug Tracker Officiel

- Bug Tracker

Flux de travail

Le flux de travail dans jasper-reports est:

1. Concevez le rapport, créez le fichier jrxml qui définit la présentation du rapport. Le jrxml peut être créé en utilisant un simple éditeur de texte, mais normalement, un IDE (JasperSoft Studio ou iReport) est utilisé à la fois pour accélérer le développement de rapports mais aussi pour avoir une vue visuelle de la mise en page.

2. Compilez le rapport (le jrxml) pour obtenir un fichier .jasper ou un objet JasperReport . Ce processus peut être comparé à un fichier `.java` en cours de compilation en `.class` .

3. Remplissez le rapport , transmettez les paramètres et une source de données au rapport pour générer l'objet d'impression JasperPrint pouvant également être enregistré dans un fichier `.jprint` .

4. Affichez, imprimez et / ou exportez le JasperPrint. Les formats d'exportation les plus courants sont les suivants: pdf, excel, word, html, cvs, etc.

Comprendre les différents groupes de rapports

Titre

Ce groupe est montré une fois au début du rapport. Il peut être utilisé comme première page en définissant l'attribut `isTitleNewPage="true"`

En-tête de page

Cela apparaît au début de chaque page en excluant la première page si la bande de titre est

utilisée et la dernière page si la bande de résumé est utilisée avec le paramètre `isSummaryWithPageHeaderAndFooter="false"`

En-tête de colonne

Cela apparaît avant la bande de détails sur chaque page.

Détail

Cette section est itérée **pour chaque enregistrement** dans la source de données fournie. Il est permis d'avoir plusieurs bandes de détails (détail 1, détail 2 .. détail n), les sont itérés comme suit

```
Row 1
    detail 1
    detail 2
    detail n
Row 2
    detail 1
    detail 2
    detail n
```

Pied de colonne

Cela apparaît sous la bande de détails sur chaque page où la bande de détail est présente. Le paramètre par défaut est la fin de la page (avant le pied de page), mais vous pouvez basculer vers la dernière bande de détails (dernier enregistrement) en définissant l'attribut `isFloatColumnFooter="true"`

Pied de page

Cela apparaît au bas de chaque page, à l'exclusion de la bande de titre, de la bande de résumé (sans le pied de page) et de la dernière bande non récapitulative si le pied de page de dernière page est utilisé.

Dernier pied de page

Cela apparaît sur la dernière page (si ce n'est pas la bande récapitulative sans le pied de page) au lieu du pied de page normal

Résumé

Cela apparaît à la fin du rapport dans la nouvelle page si `isSummaryNewPage="true"` est défini et avec

l'en-tête et le pied de page si `isSummaryWithPageHeaderAndFooter="true"`

En-tête de groupe

Cette section apparaît si un groupe est défini chaque fois que l'expression du groupe change, avant la bande de détail.

Group Footer

Cette section apparaît si un groupe est défini chaque fois *avant que* l'expression de groupe ne change, après la bande de détail.

Contexte

Ce groupe est affiché sur chaque page en arrière-plan de toutes les autres bandes.

Pas de données

Cela apparaît uniquement si aucune source de données n'a été transmise ou si la source de données est vide (0 enregistrement) et que `whenNoDataType="NoDataSection"` est défini.

Formats de fichiers de rapport Jasper

- `.jrxml` est le fichier de conception de rapport, son format est en XML lisible par l'homme, il peut être intégré dans un objet `JasperReport` et enregistré en tant que `.jasper`
- `.jasper` est la version compilée du `.jrxml` et peut être chargé directement dans un objet `JasperReport` prêt à être rempli avec des données.
- `.jrprint` est l'objet `JasperPrint` sérialisé, un rapport qui a déjà été rempli de données et peut être chargé pour être imprimé, affiché et / ou exporté au format souhaité.
- `.jrpxml` est le rappresentativo XML d'un objet `JasperPrint` , il peut être modifié puis désarchivé pour récupérer l'objet `JasperPrint`

Lire Démarrer avec les rapports Jasper en ligne: https://riptutorial.com/fr/jasper-reports/topic/3594/demarrer-avec-les-rapports-jasper

Chapitre 2: Compilez JasperReports .jrxml à .jasper

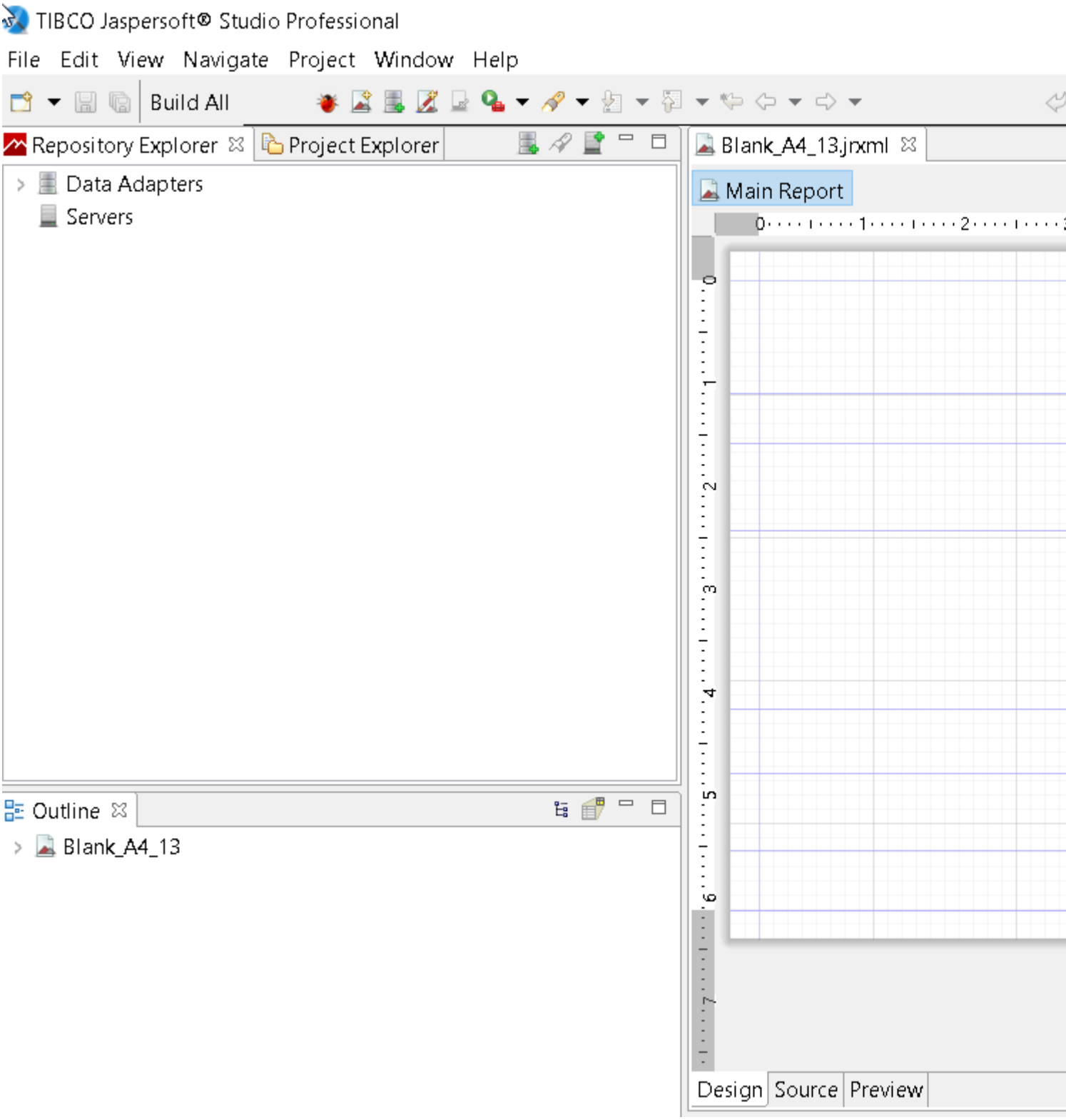
Examples

Avec IDE (environnement de développement intégré)

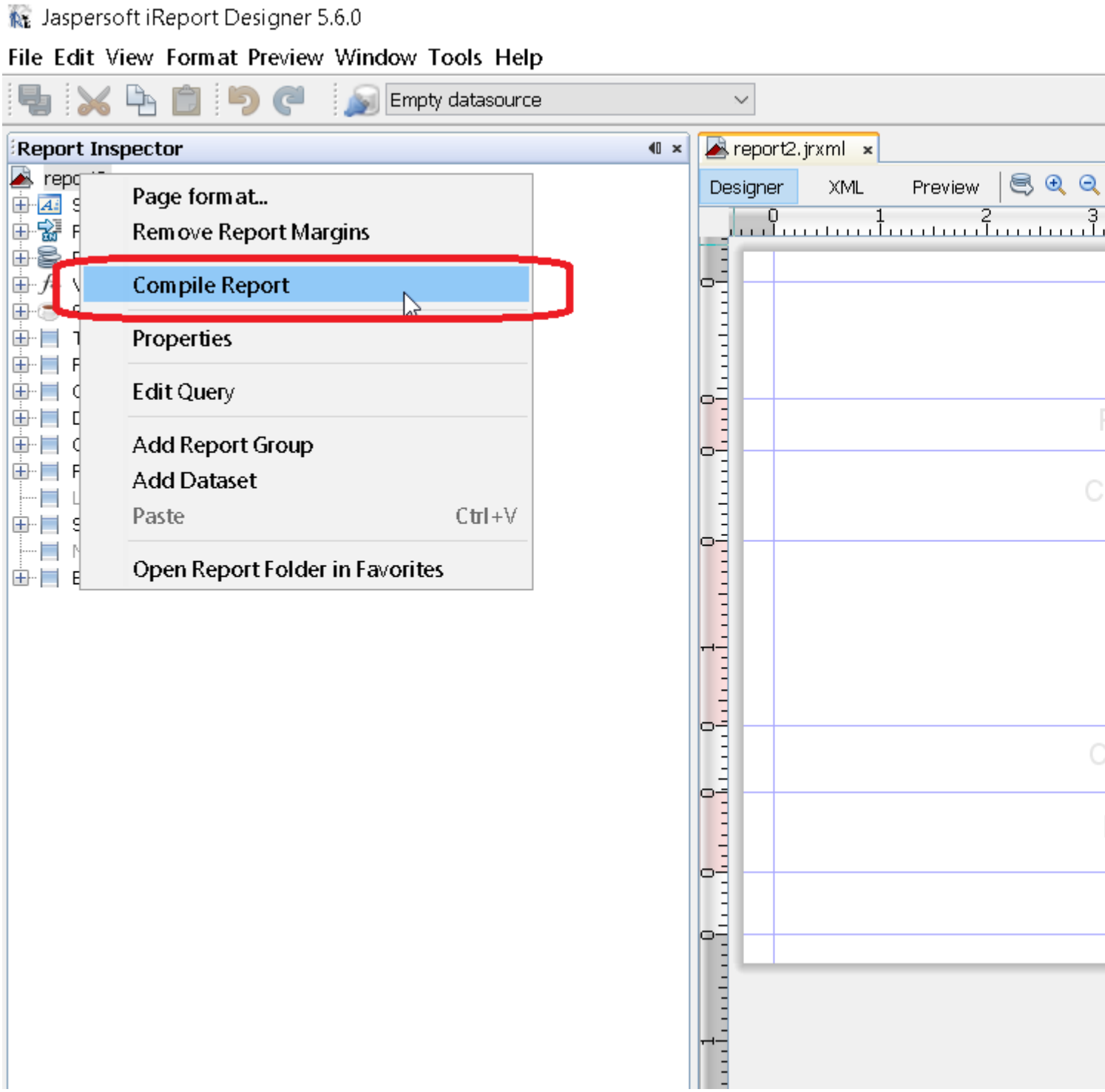
Dans IDE Jaspersoft Studio (*JSS*) ou l'ancienne version de iReport Designer, il suffit d'appuyer sur **Aperçu** .

Le fichier de conception JasperReports `.jrxml` sera automatiquement compilé en `.jasper` dans le même dossier que `.jrxml` si **aucune erreur** n'est présente.

Une autre façon est d'appuyer *sur le* bouton *"Compiler le rapport"* dans *JSS*



ou utilisez le menu contextuel *"Compile Report"* appelé depuis *Report Inspector* dans *iReport*



Avec Apache Ant

```
<target name="compile" description="Compiles report designs specified using the 'srcdir' in
the &lt;jrc&gt; tag." depends="prepare-compile-classpath">
  <mkdir dir="./build/reports"/>
    <taskdef name="jrc" classname="net.sf.jasperreports.ant.JRAntCompileTask">
        <classpath refid="project-classpath"/>
    </taskdef>
    <jrc
            srcdir="./reports"
            destdir="./build/reports"
            tempdir="./build/reports"
            keepjava="true"
```

```
            xmlvalidation="true">
        <classpath refid="sample-classpath"/>
        <include name="**/*.jrxml"/>
    </jrc>
</target>
```

L'outil de compilation Apache Ant doit être correctement installé sur votre système

Avec java

Bien qu'il soit possible de compiler des fichiers `.jrxml` dans des fichiers `.jasper` en utilisant du code Java, cela engendre un `.jasper` performance qu'il vaut mieux éviter en pré-compilant des fichiers `.jrxml` à l'aide de l'EDI. Dans cette optique, la compilation des fichiers `.jrxml` peut être effectuée à l'aide de JasperCompileManager comme suit:

```
JasperCompileManager.compileReportToFile(
            "designFile.jrxml", //Relative or absoulte path to the .jrxml file to compile
            "compiled.jasper"); //Relative or absolute path to the compiled file .jasper
```

Avec Apache Maven

Le *plugin JasperReports* par Alex Nederlof est une bonne alternative au plugin org.codehaus.mojo: jasperreports-maven-plugin abandonné.

L'ajout de plugin est une procédure simple et typique:

```
<build>
    <plugins>
        <plugin>
            <groupId>com.alexnederlof</groupId>
            <artifactId>jasperreports-plugin</artifactId>
            <version>2.3</version>
            <executions>
                <execution>
                    <phase>process-sources</phase>
                    <goals>
                        <goal>jasper</goal>
                    </goals>
                </execution>
            </executions>
            <configuration>
                <sourceDirectory>src/main/resources/jrxml</sourceDirectory>
                <outputDirectory>${project.build.directory}/jasper</outputDirectory>
            </configuration>
        </plugin>
    </plugins>
</build>
```

La commande de compilation avec *Maven* :

mvn jasperreports: jasper

Les fichiers *jasper* seront créés dans le dossier *$ {project.build.directory} / jasper* (par exemple,

dans / *target* / *jasper*)

Lire Compilez JasperReports .jrxml à .jasper en ligne: https://riptutorial.com/fr/jasper-reports/topic/4943/compilez-jasperreports--jrxml-a--jasper

Chapitre 3: Exporter en pdf

Remarques

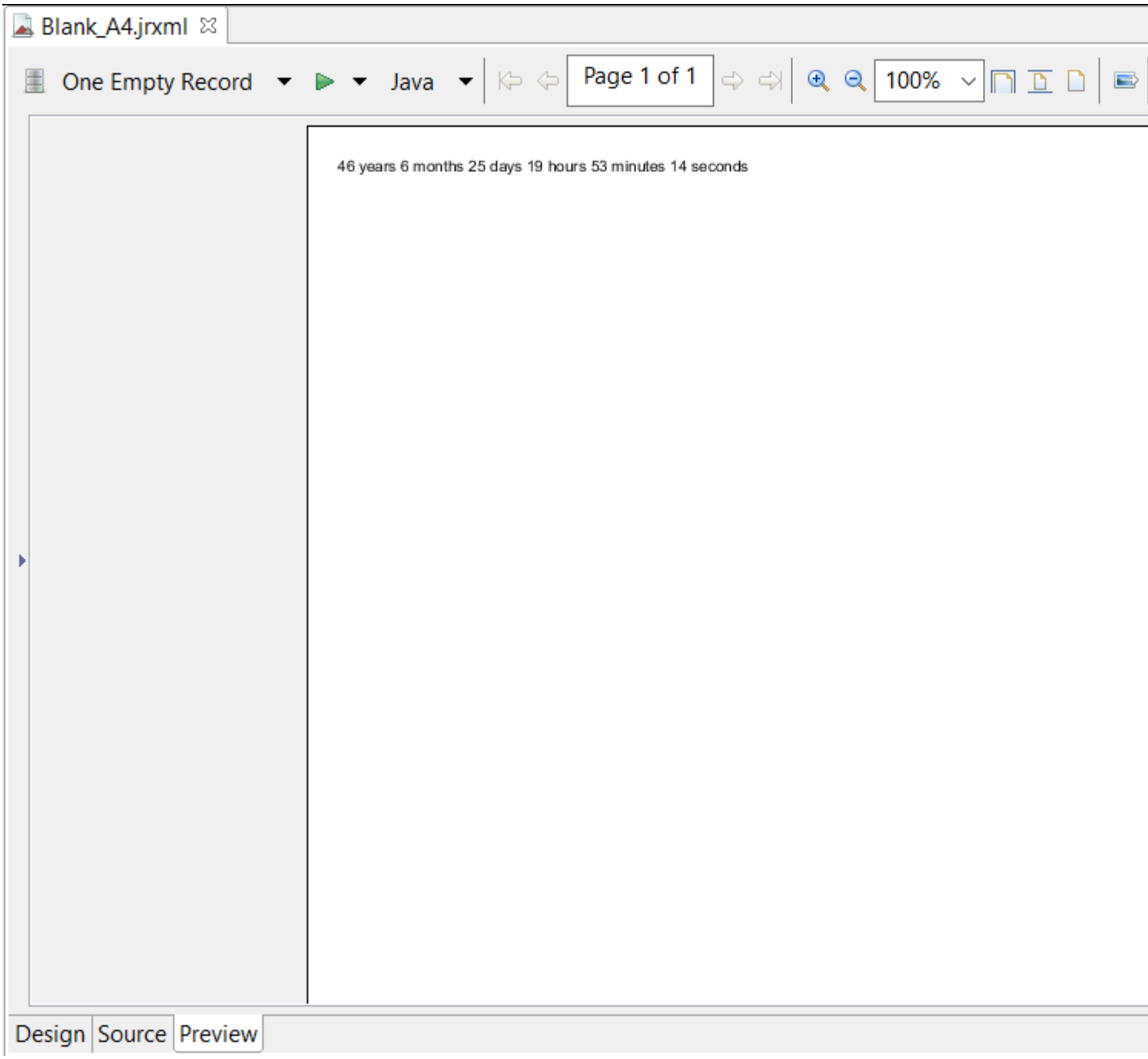
Pour rendre les **polices** correctement en pdf, les polices doivent toujours être utilisées (dans classpath)

Examples

Avec IDE (environnement de développement intégré)

JasperSoft Studio

En mode Aperçu, exécutez le rapport en cliquant sur la flèche verte. Si aucune erreur ne s'est produite dans le menu d'exportation, cliquez sur le bouton d'exportation (image disque) et sélectionnez «Exporter en format PDF».



Avec java

Pour exporter un objet, vous devez remplir le rapport pour obtenir l'objet JasperPrint .

Exporter un fichier JasperPrint unique (jrxml unique) vers un fichier

```
// 1. Create exporter instance
JRPdfExporter exporter = new JRPdfExporter();

// 2. Set exporter input document
exporter.setExporterInput(new SimpleExporterInput(jasperPrint));
```

```
// 3. Set file path for exporter output
exporter.setExporterOutput(new SimpleOutputStreamExporterOutput("/path/filename.pdf"));

// 4. Create configuration instance
SimplePdfExporterConfiguration configuration = new SimplePdfExporterConfiguration();

// 5. Associate configuration with exporter
exporter.setConfiguration(configuration);

// 6. Fill export and write to file path
exporter.exportReport();
```

Exporter plusieurs fichiers JasperPrint (multiples jrxml) vers un fichier unique

Seules les premières étapes diffèrent de la précédente:

```
List<JasperPrint> jasperPrintList = new ArrayList<>();
jasperPrintList.add(jasperPrint1);
jasperPrintList.add(jasperPrint2);

JRPdfExporter exporter = new JRPdfExporter();
exporter.setExporterInput(SimpleExporterInput.getInstance(jasperPrintList));
```

Les étapes restantes sont les mêmes:

```
exporter.setExporterOutput(new SimpleOutputStreamExporterOutput("/path/filename.pdf"));
SimplePdfExporterConfiguration configuration = new SimplePdfExporterConfiguration();
exporter.setConfiguration(configuration);
exporter.exportReport();
```

Voir API SimplePdfExporterConfiguration pour les détails de configuration.

Lire Exporter en pdf en ligne: https://riptutorial.com/fr/jasper-reports/topic/4190/exporter-en-pdf

Chapitre 4: Exporter vers xls / xlsx

Examples

Avec java

Exporter au format *xlsx*

```
try (InputStream inputStream = JRLoader.getResourceInputStream(path)) {  // read report as
input stream
    JasperReport  jasperReport =
JasperCompileManager.compileReport(JRXmlLoader.load(inputStream)); // compile report

    Map<String, Object> params = new HashMap<>(); // init map with report's parameters
    params.put(JRParameter.REPORT_LOCALE, Locale.US);
    params.put(JRParameter.IS_IGNORE_PAGINATION, true);
    JasperPrint jasperPrint = JasperFillManager.fillReport(jasperReport, params, connection);
// prepare report - passs parameters and jdbc connection

    JRXlsxExporter exporter = new JRXlsxExporter(); // initialize exporter
    exporter.setExporterInput(new SimpleExporterInput(jasperPrint)); // set compiled report as
input
    exporter.setExporterOutput(new SimpleOutputStreamExporterOutput(destFile));  // set output
file via path with filename
    SimpleXlsxReportConfiguration configuration = new SimpleXlsxReportConfiguration();
    configuration.setOnePagePerSheet(true); // setup configuration
    configuration.setDetectCellType(true);
    exporter.setConfiguration(configuration); // set configuration
    exporter.exportReport();
}
```

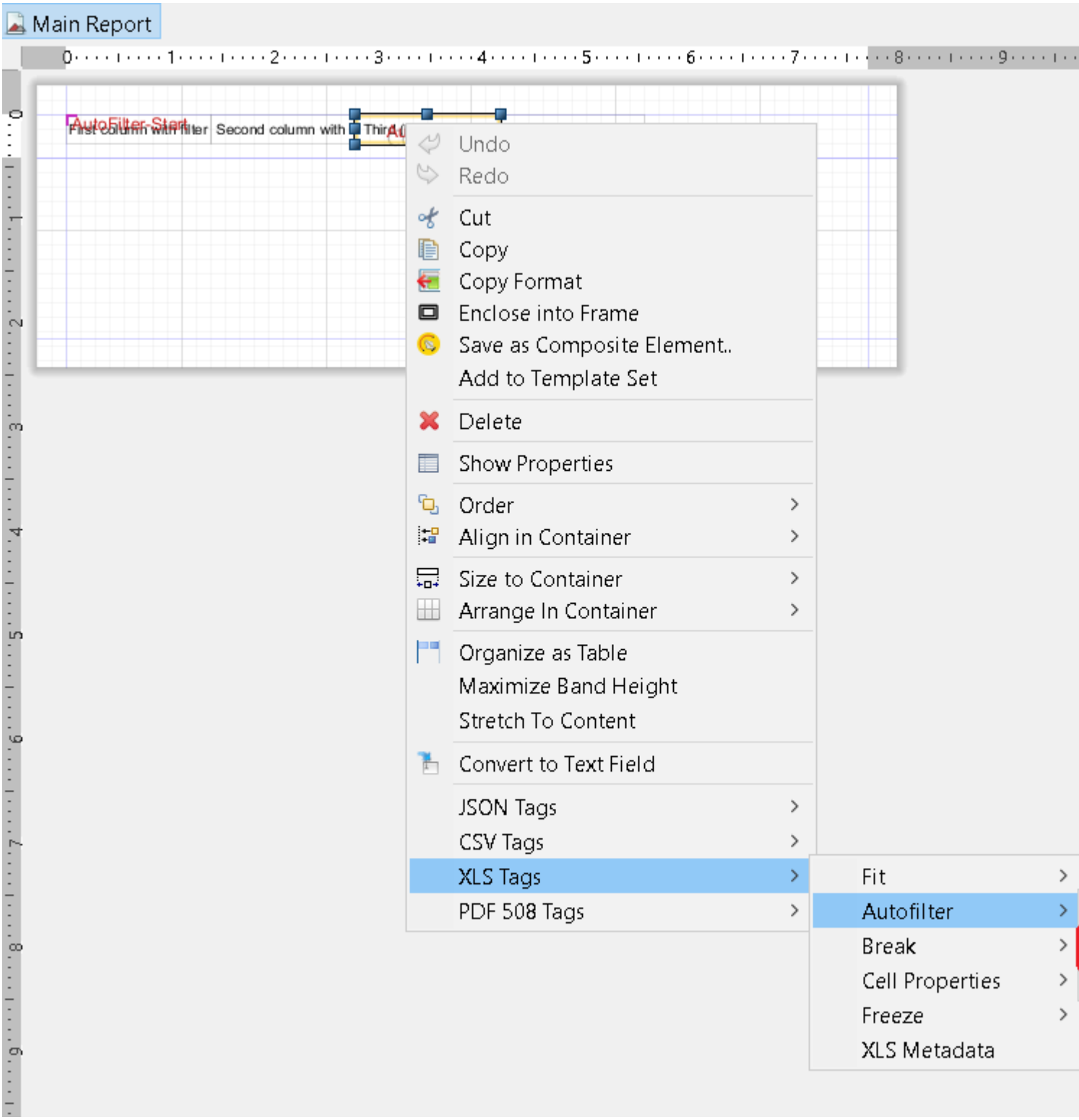
Ajout de filtre automatique pour les colonnes

L'utilisation de la propriété ***net.sf.jasperreports.export.xls.auto.filter*** permet d'ajouter un filtre automatique dans le fichier xls généré.

```
<columnHeader>
    <band height="30" splitType="Stretch">
        <staticText>
            <reportElement x="0" y="0" width="100" height="20">
                <property name="net.sf.jasperreports.export.xls.auto.filter" value="Start"/>
            </reportElement>
            <text><![CDATA[First column with filter]]></text>
        </staticText>
        <staticText>
            <reportElement x="100" y="0" width="100" height="20"/>
            <text><![CDATA[Second column with filter]]></text>
        </staticText>
        <staticText>
            <reportElement x="200" y="0" width="100" height="20">
                <property name="net.sf.jasperreports.export.xls.auto.filter" value="End"/>
            </reportElement>
            <text><![CDATA[Third (Last) column with filter]]></text>
        </staticText>
```

```
        <staticText>
            <reportElement x="300" y="0" width="100" height="20"/>
            <text><![CDATA[Fourth column without filter]]></text>
        </staticText>
    </band>
</columnHeader>
```

La propriété peut être définie dans *Jaspersoft Studio* à l'aide du menu contextuel ou manuellement en modifiant le fichier *jrxml* .



Lire Exporter vers xls / xlsx en ligne: https://riptutorial.com/fr/jasper-reports/topic/5008/exporter-

vers-xls---xlsx

Chapitre 5: Extensions de polices

Examples

Création et utilisation d'extensions de polices

Créez une extension de police à l'aide de l'EDI. Consultez la documentation de iReport ou de Jaspersoft Studio pour plus de détails. L'extension de police peut également être créée manuellement .

Que sont les extensions de police?

En utilisant un `textElement` vous pouvez spécifier une police (si aucune police par défaut n'est spécifiée, `SansSerif` est utilisé)

```
<textElement>
    <font fontName="DejaVu Sans"/>
</textElement>
```

Pour calculer la métrique de la police (pour les sauts de ligne, l'alignement, etc.) et rendre la police correctement, la **police** doit être **mappée dans la machine** virtuelle Java (Java virtual macchine). Vous pouvez installer le fichier de police directement sur la machine virtuelle Java, mais cela n'est pas encourageant.

Du Guide JasperReport Ultimate:

> Nous encourageons fortement les utilisateurs à utiliser uniquement des polices dérivées d'extensions de polices, car c'est le seul moyen de s'assurer que les polices seront disponibles pour l'application lorsque les rapports seront exécutés au moment de l'exécution. L'utilisation de polices système entraîne toujours le risque que les rapports ne fonctionnent pas correctement lorsqu'ils sont déployés sur une nouvelle machine sur laquelle ces polices ne sont peut-être pas installées

Extension de police par défaut

JasperReports fournit une extension de police par défaut (voir la distribution de maven jasperreports-fonts.jar). En ajoutant ceci à classpath, vous pouvez utiliser le fontName suivant sans créer votre propre extension de police

> DejaVu Sans
> DejaVu Serif
> DejaVu Sans Mono

Problèmes courants

Problèmes à prendre en compte lors de l'utilisation des polices en pdf (itext):

- Lors de l'exportation au format PDF, si le texte n'est pas rendu correctement (pièces manquantes, caractères non affichés, ne pas s'afficher ou dimensionnés correctement), les **extensions de police** sont probablement manquantes.
- Le `.tff` **pris en charge** (OpenType) et la police peut-elle **rendre** le caractère? Toutes les polices ne rendent pas tous les caractères dans `UTF-8` .
- Le **codage correct est-il** transmis à iText? En cas de doute (ou en général), utilisez le **codage** `Identity-H` est recommandé pour les nouvelles normes PDF et vous permet de combiner différents encodages.
- La police **intégrée permet** -elle à un PDF partagé sur plusieurs ordinateurs d'afficher le contenu même si la police n'est pas installée? Si la police ne fait pas partie des 14 polices standard de type 1, elles l'incluent toujours.

Notez que la version de iText utilisée par le rapport Jasper ne rendra pas toutes les polices (problème de ligaturation). Vous pouvez tester directement la police et l'encodage `ttf` .

Lire Extensions de polices en ligne: https://riptutorial.com/fr/jasper-reports/topic/5773/extensions-de-polices

Chapitre 6: Remplir le rapport

Paramètres

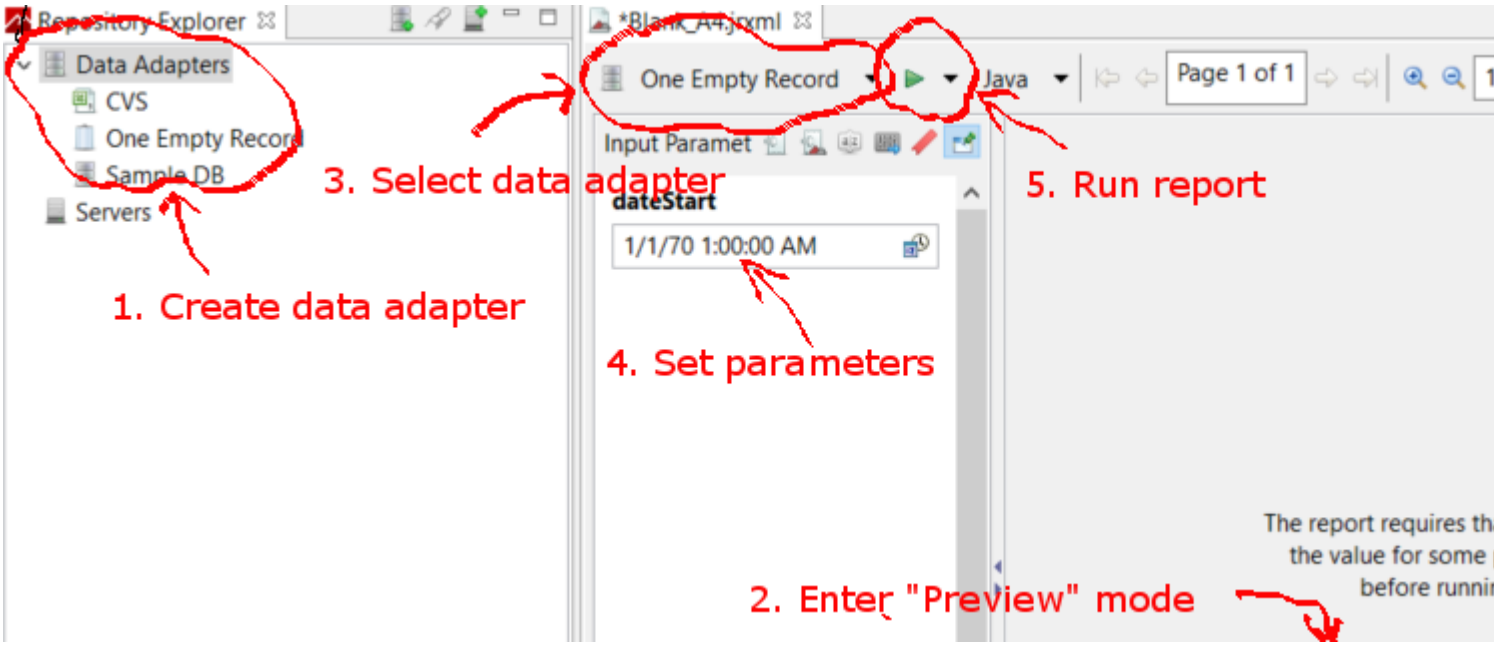
Paramètres	Colonne
jasperPrint	Le résultat du processus de remplissage pouvant être exporté au format souhaité
reportTemplate	Le fichier de conception compilé `.jasper`
paramètres	Le paramètre Map , qui, s'il est défini, peut être une référence à l'intérieur du rapport par `$P{key}`
la source de données	Un net.sf.jasperreports.engine.JRDataSource
connexion	Une connexion à la base de données java.sql.Connection

Examples

Avec IDE (environnement de développement intégré)

JasperSoft Studio

1. Si la connexion à la source de données ou à la base de données est nécessaire pour remplir le rapport, créez votre adaptateur de données dans l'Explorateur de référentiels en cliquant avec le bouton droit sur "Adaptateurs de données" en sélectionnant "Créer un adaptateur de données".
2. Passez en mode aperçu en sélectionnant l'onglet **Aperçu** (aucune erreur de conception ne doit être présente)
3. Sélectionnez la source de données souhaitée (si aucune source de données n'est requise, sélectionnez "One Empty Record"
4. Définir le paramètre comme souhaité
5. Remplir le rapport en cliquant sur la flèche verte "Exécuter le rapport"



Remplir le modèle JasperReport avec Java

Exigences communes

Tous les rapports, quel que soit le mode de présentation des données, empruntent un chemin vers le modèle de rapport et une mappe de paramètres. Les variables sont utilisées dans tous les exemples suivants:

```
// Parameters passed into the report.
Map<String, Object> parameters = new HashMap<>();

// Arbitrary parameter passed into the report.
parameters.put("KEY", "Value");

// The compiled report design.
String path = "path/to/template.jasper";
```

L'utilisation d'un fichier `.jrxml` entraîne une étape de compilation supplémentaire inutile dans la plupart des cas. Sauf si vous avez écrit un logiciel personnalisé pour modifier le `.jrxml` avant l'`.jrxml` du rapport (par exemple, l'ajout ou la suppression dynamique de colonnes), utilisez le fichier `.jasper` comme indiqué dans les exemples suivants.

Utilisation d'une connexion à une base de données

```
// Establish a database connection.
Connection connection = DriverManager.getConnection(url, username, password);

// Fill the report, get the JasperPrint that can be exported to desired format.
JasperPrint jasperPrint = JasperFillManager.fillReport(
```

```
  path, parameters, connection);
```

Utiliser une source de données personnalisée

```
// Populate this list of beans as per your requirements.
List<Bean> beans = new ArrayList<>();

// Wrap the beans in a beans in a JRBeanCollectionDataSource.
JRBeanCollectionDataSource datasource = new JRBeanCollectionDataSource(beans);

// Fill the report, get the JasperPrint that can be exported to desired format.
JasperPrint jasperPrint = JasperFillManager.fillReport(
  path, parameters, datasource);
```

Sans source de données, bande de détails inutilisée

```
// Fill the report, get the JasperPrint that can be exported to desired format.
JasperPrint jasperPrint = JasperFillManager.fillReport(path, parameters);
```

> Sans l'attribut de données, l'attribut `whenNoDataType="AllSectionsNoDetail"` de l'élément `JasperReport` doit être défini, sinon un rapport vide (vide) sera généré.

Lire Remplir le rapport en ligne: https://riptutorial.com/fr/jasper-reports/topic/3958/remplir-le-rapport

Chapitre 7: Utiliser des sous-rapports

Paramètres

Paramètre	Détails
parametersMapExpression	La carte avec les paramètres. *Non requis*
sous-rapportParamètre	La paire de nom et de valeur (définie avec ***subreportParameterExpression***). *Non requis* Plusieurs paramètres peuvent être transmis au sous-rapport
connectionExpression	Connexion pour obtenir des données. *Non requis*
dataSourceExpression	Expression pour transmettre la source de données. *Non requis*
sous-rapportExpression	Le chemin d'accès / URI du sous-rapport ou même l'objet JasperReport. *Non requis*
returnValue	La paire de nom et de valeur. *Non requis* Plusieurs valeurs peuvent être renvoyées du sous-rapport au rapport principal

Remarques

- Les sous-rapports peuvent être utilisés pour créer des rapports complexes. La réutilisation des rapports existants est un autre objectif de l'utilisation des sous-rapports.
- Le sous-rapport sera affiché dans le cadre du rapport maître en cas d'utilisation de l'élément `<subreport>` .
- La valeur du paramètre ***subreportExpression*** est différente pour l'utilisation sur *JasperReports Server* ou simplement par le framework *JasperReports* (certaines *API* utilisant ou utilisant IDE).

  Pour *JasperReports Server,* cela ressemble à:

  ```
  <subreportExpression><![CDATA["repo:subreport.jrxml"]]></subreportExpression>
  ```

  Pour utiliser uniquement le moteur *JasperReports* :

  ```
  <subreportExpression><![CDATA["/somePath/subreport.jasper"]]></subreportExpression>
  ```

  La grande explication de *@AndreasDietrich* peut être trouvée sur *JasperServer: Impossible de localiser le* message d' *exception de sous-rapport*

- Pour certaines raisons, le sous-rapport peut être utilisé comme rapport commun - sans appeler depuis le rapport principal (avec l'aide de l'élément `<subreport>`). Le sous-rapport est toujours un rapport.

Examples

Passer la connexion au sous-rapport; retourne les valeurs au rapport principal

Ceci est un extrait du rapport principal. Deux paramètres et la connexion (par exemple, *jdbc*) passent au sous-rapport. Une valeur est renvoyée du sous-rapport au rapport principal, cette valeur (*variable*) peut être utilisée dans le rapport principal

```
<subreport>
    <reportElement x="0" y="80" width="200" height="100"/>
    <subreportParameter name="someSubreportParameter">

<subreportParameterExpression><![CDATA[$P{someMasterReportParamter}]]></subreportParameterExpression>

    </subreportParameter>
    <subreportParameter name="anotherSubreportParameter">
        <subreportParameterExpression><![CDATA["Some text - constant
value"]]></subreportParameterExpression>
    </subreportParameter>
    <connectionExpression><![CDATA[$P{REPORT_CONNECTION}]]></connectionExpression>
    <returnValue subreportVariable="someVariableInSubreport"
toVariable="someVariableInMasterReport"/>
    <subreportExpression><![CDATA["$P{SUBREPORT_DIR} +
"subreport.jasper"]]></subreportExpression>
</subreport>
```

Transmission de données au sous-rapport

Ceci est un extrait du rapport principal. La source de données est transmise au sous-rapport à l'aide du constructeur ***net.sf.jasperreports.engine.data.JRBeanCollectionDataSource***

```
<field name="someFieldWithList" class="java.util.List"/>
<!-- ...... -->
<subreport>
    <reportElement x="0" y="0" width="200" height="70"/>
    <parametersMapExpression><![CDATA[$P{REPORT_PARAMETERS_MAP}]]></parametersMapExpression>

<dataSourceExpression><![CDATA[net.sf.jasperreports.engine.data.JRBeanCollectionDataSource($F{someFiel

    <subreportExpression><![CDATA[$P{SUBREPORT_DIR} +
"subreport.jasper"]]></subreportExpression>
</subreport>
```

Lire Utiliser des sous-rapports en ligne: https://riptutorial.com/fr/jasper-reports/topic/5452/utiliser-des-sous-rapports

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec les rapports Jasper	Alex K, Community, Dave Jarvis, Petter Friberg
2	Compilez JasperReports .jrxml à .jasper	Alex K, Dave Jarvis, Petter Friberg
3	Exporter en pdf	Alex K, Dave Jarvis, Petter Friberg, RamenChef
4	Exporter vers xls / xlsx	Alex K
5	Extensions de polices	Dave Jarvis, Petter Friberg
6	Remplir le rapport	Alex K, Dave Jarvis, Petter Friberg
7	Utiliser des sous-rapports	Alex K