FREE eBook

LEARNING jasper-reports

#jasper-reports

Table of Contents

About

Chapter 1: Getting started with jasper-reports

Remarks

There are several libraries used *JasperReports Java API* for creating reports with Java:

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

This libraries/frameworks can build reports "on fly" with or without using report's template (*jrxml* file)

Versions

JasperReports library

Version	Release date
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
3.7.5	2010-09-22
3.7.0	2009-12-08

Version	Release date
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 for designing reports

The current version of designer is based on *Eclipse*: *Jaspersoft Studio*.

The previous version of designer was based on *NetBeans*: *iReport Desigher*.

The first version of *iReport Designer* was independent application - *iReport Classic*

Examples

Installation or Setup

JasperReports Library

JasperReports is a open source Java based reporting tool. The *JasperReports* Library can be downloaded from the Jaspersoft Community for the latest release.

> In recent releases the third-party jars in the lib folder are **not** distributed, they need to be download from public repositories, see distributed `pom.xml` for dependencies. Maven can be used to retrieve all dependencies including the transient ones in the target/dependence folder.

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

Jaspersoft Studio (IDE)

Jaspersoft Studio is the official design client for JasperReports--built on the Eclipse platform--to replace iReport Designer.

iReport Designer (IDE)

iReport Designer is the previous report designer for JasperReports. Version 5.6.0 (released in May of 2014) was the last official version; vendor support ended at the end of 2015.

JasperReport Commuity resources

JasperReports Library FAQs

- FAQ

Source code

- JasperReports Library source code

Tutorials

- Tutorials Point
- JasperReports Ultimate Guide

Samples

- Sample Reference

References

- Official documentation
- Community Wiki

Official Bug Tracker

- Bug Tracker

Work flow

The work flow in jasper-reports is:

1. Design the report, create the jrxml file that defines the report layout. The jrxml can be create by using a simple texteditor but normally an IDE (JasperSoft Studio or iReport) is used both to speed up report development but also to have a visual view of layout.

2. Compile the report (the jrxml) to get a .jasper file or a JasperReport object. This process can be compared with a `.java` file being compiled to `.class`.

3. Fill the report, pass parameters and a datasource to the report to generate the print object JasperPrint that can also be saved to a `.jprint` file

4. View, print and/or export the JasperPrint. The most commons export format are supported as pdf, excel, word, html, cvs etc.

Understanding the different report bands

Title

This band is showed once at the beginning of the report. It can be used as first page by setting the attribute `isTitleNewPage="true"`

Page Header

This appears at the beginning of each page excluding first page if Title band is used and last page if Summary band is used with setting `isSummaryWithPageHeaderAndFooter="false"`

Column Header

This appears before the detail band on each page.

Detail

This section is iterated **for each record** in datasource supplied. It is allowed to have multiple detail band (detail 1, detail 2 .. detail n), the are iterated as follows

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

Column Footer

This appears below the detail band on each page where detail band is present. The default setting is end of page (before Page footer) but this can be switch to under last detail band (last record) by setting the attribute `isFloatColumnFooter="true"`

Page Footer

This appears at the bottom of each page excluding title band, summary band (without page footer) and last non summary band if Last Page Footer is used.

Last Page Footer

This appears on last page (if not summary band without page footer) instead of normal Page Footer

Summary

This appears at the end of the report in new page if `isSummaryNewPage="true"` is set and with page header and footer if `isSummaryWithPageHeaderAndFooter="true"`

Group Header

This section appears if a group is defined every time the group expression change, before the detail band.

Group Footer

This section appears if a group is defined every time *before* the group expression change, after the detail band.

Background

This band is displayed on every page as background to all other bands.

No Data

This appears only if no datasource was passed or the datasource is empty (0 records) and `whenNoDataType="NoDataSection"` is set.

Jasper report file formats

- `.jrxml` is the report design file, it's format is in human readable XML, it can be complied into a `JasperReport` object and saved as a `.jasper`
- `.jasper` is the compiled version of the `.jrxml` and can be loaded directly into a `JasperReport` object ready to be filled with data
- `.jrprint` is the serialized `JasperPrint` object, a report that have already been filled with data and can be loaded to be printed, viewed and/or exported to desired format.
- `.jrpxml` is the XML rappresentativo of a `JasperPrint` object it can be modified and then unmarshaled to retrieve the `JasperPrint` object

Read Getting started with jasper-reports online: https://riptutorial.com/jasper-reports/topic/3594/getting-started-with-jasper-reports

Chapter 2: Compile JasperReports .jrxml to .jasper

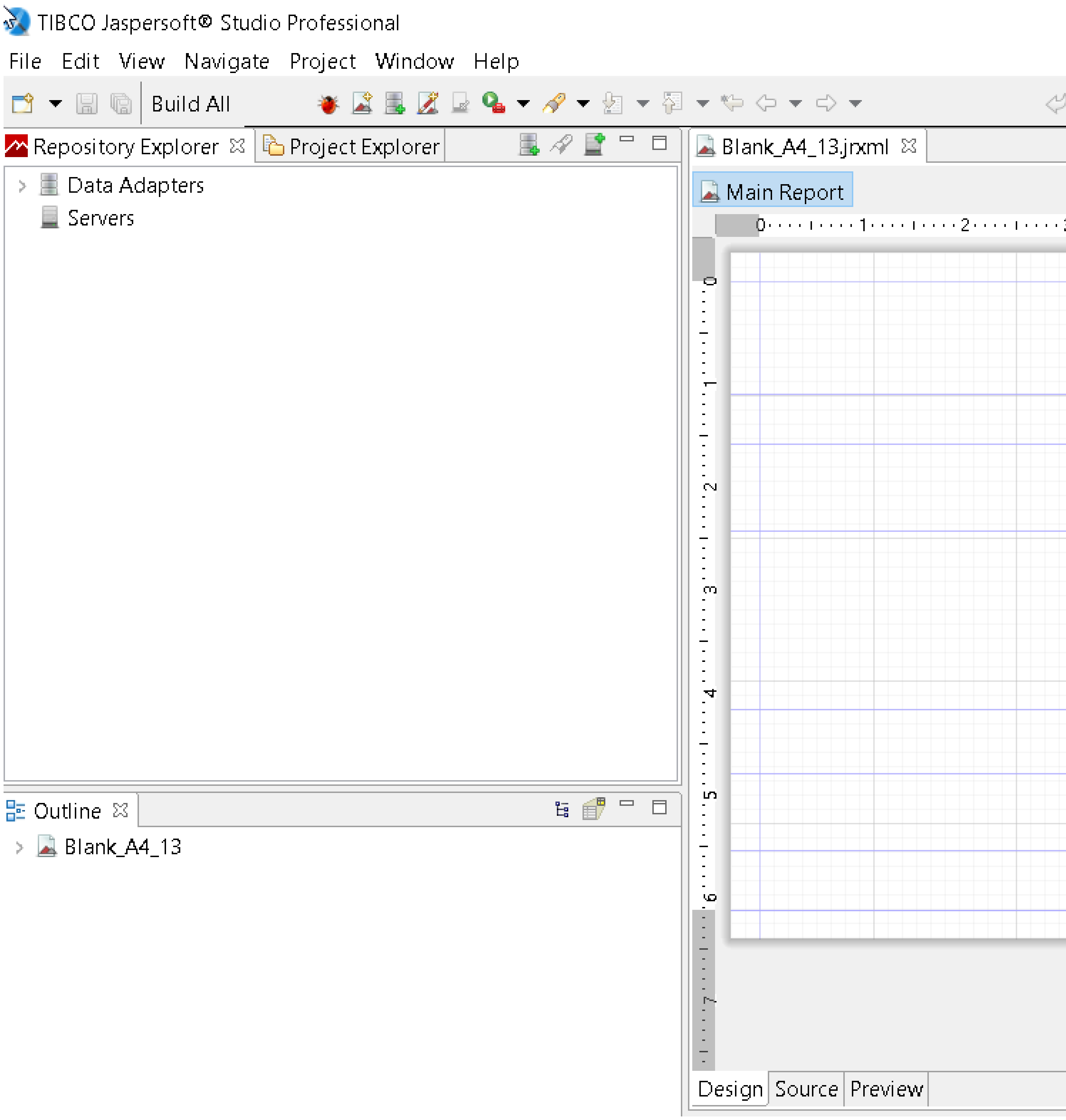
Examples

With IDE (Integrated development environment)

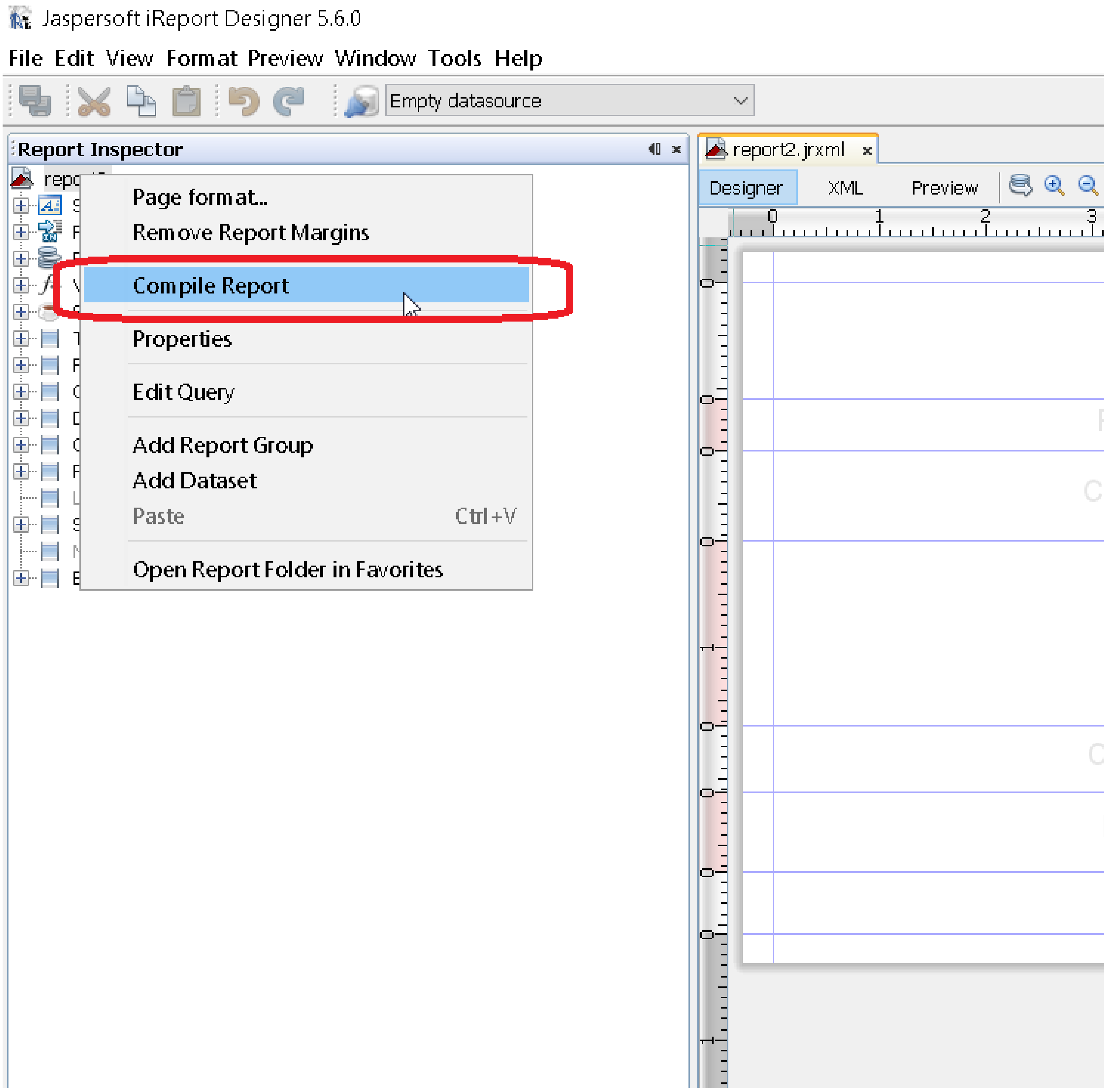
In IDE Jaspersoft Studio (*JSS*) or the older version iReport Designer it is sufficient to press **Preview**.

The JasperReports design file `.jrxml` will automatically be compiled to `.jasper` in same folder as `.jrxml` if **no errors** are present.

Another way is to press *"Compile Report"* button in *JSS*



or use the context menu *"Compile Report"* called from *Report Inspector* in *iReport*



With 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>
```

Apache Ant build tool needs to be correctly installed on your system

With Java

While it is possible to compile `.jrxml` files into `.jasper` files using Java code, this incurs a performance hit that is best avoided by pre-compiling `.jrxml` files using the IDE. With that in mind, compiling `.jrxml` files can be accomplished using the JasperCompileManager as follows:

```
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
```

With Apache Maven

The *JasperReports-plugin* by Alex Nederlof is a good alternative of abandoned org.codehaus.mojo:jasperreports-maven-plugin plugin.

The adding of plugin is a typical, simple procedure:

```
<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>
```

The command for compilation with *Maven*:

mvn jasperreports:jasper

The *jasper* files will be created in *${project.build.directory}/jasper* folder (for example, in */target/jasper*)

Read Compile JasperReports .jrxml to .jasper online: https://riptutorial.com/jasper-reports/topic/4943/compile-jasperreports--jrxml-to--jasper

Chapter 3: Export to pdf

Remarks

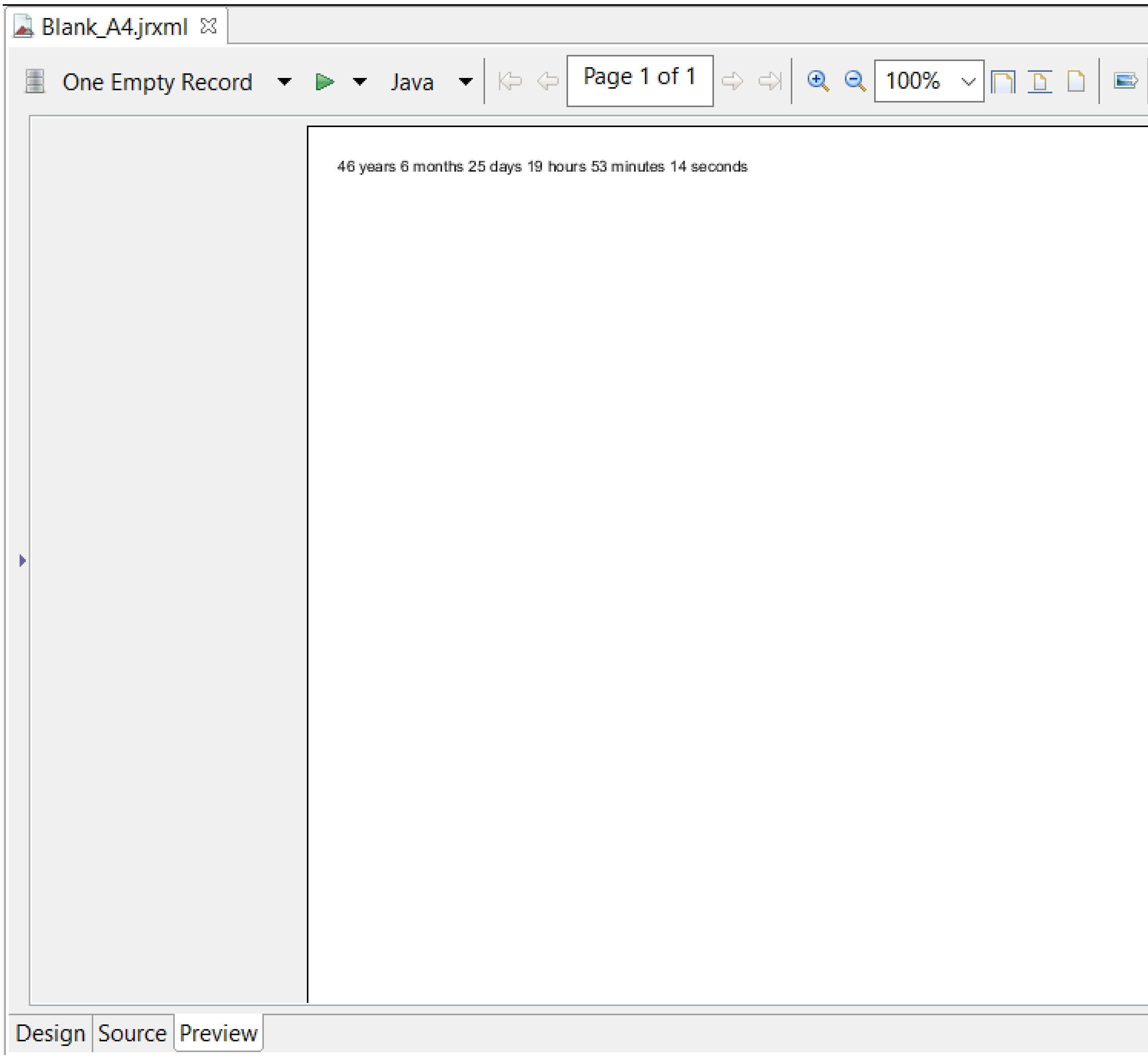
To render **fonts** correctly in pdf font-extensions should always be used (in classpath)

Examples

With IDE (Integrated development environment)

JasperSoft Studio

In Preview, run report by clicking green arrow, if no errors the export menu will be enable, click the export button (disk image) and select "Export As Pdf"



46 years 6 months 25 days 19 hours 53 minutes 14 seconds

With Java

To export a you need to fill the report to get the JasperPrint object.

Export single JasperPrint (single jrxml) to file

```
// 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();
```

Export multiple JasperPrint's (multiple jrxml) to single file

Only the first steps differ from the previous set:

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

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

The remaining steps are the same:

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

See SimplePdfExporterConfiguration API for configuration details.

Read Export to pdf online: https://riptutorial.com/jasper-reports/topic/4190/export-to-pdf

Chapter 4: Export to xls/xlsx

Examples

With Java

Export to *xlsx* format

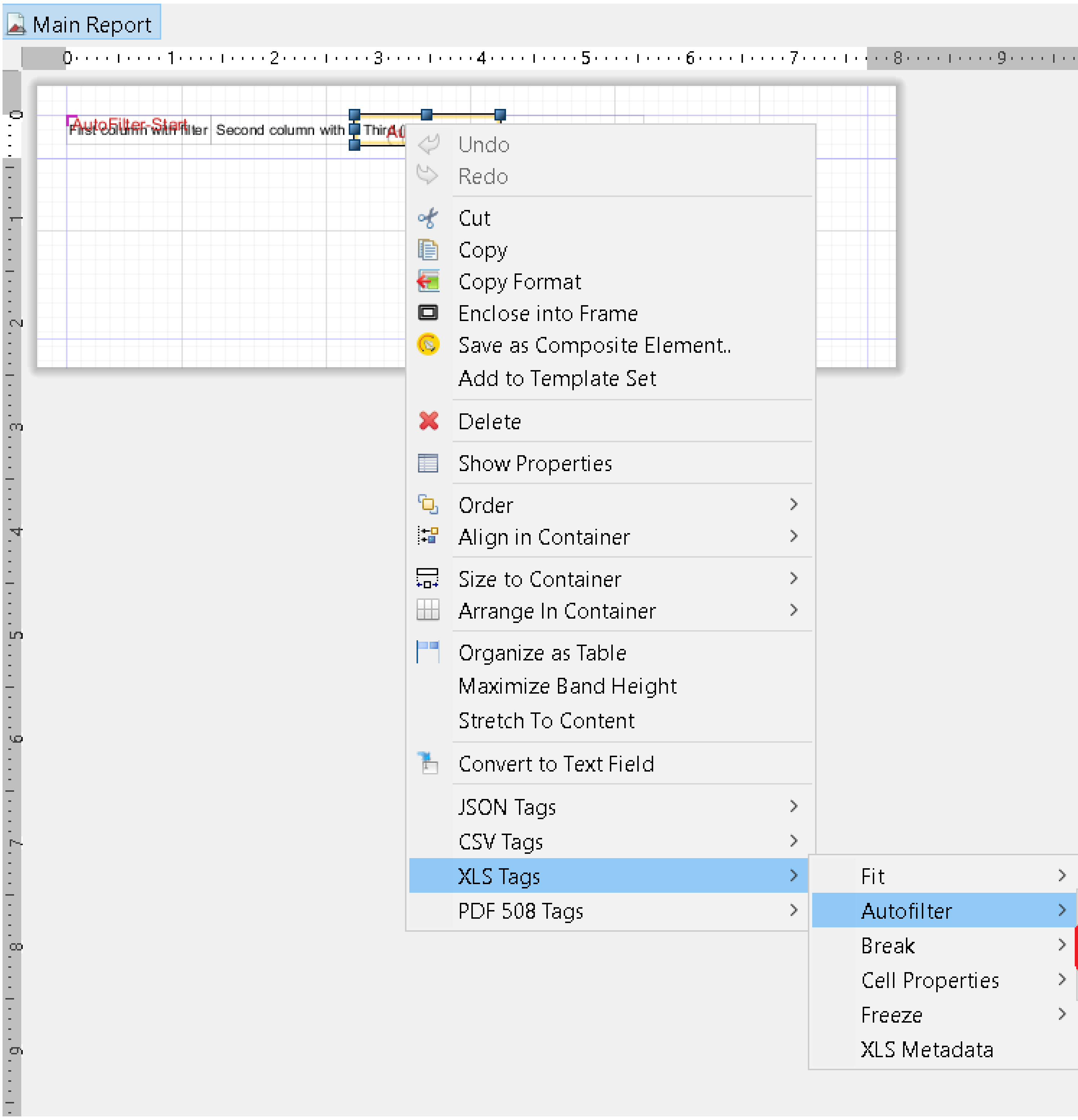
```
try (InputStream inputStream = JRLoader.getResourceInputStream(path)) {  // read report as
input stream
    JasperReport  jasperReport =
JasperCompileManager.compileReport(JRXmlLoader.load(inputStream)); // compile report

    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();
}
```

Adding autofilter for columns

The using of ***net.sf.jasperreports.export.xls.auto.filter*** property allow to add autofilter in generated xls file.

```
<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>
```

The property can be set in *Jaspersoft Studio* with help of context menu or manually by editing *jrxml* file.



Read Export to xls/xlsx online: https://riptutorial.com/jasper-reports/topic/5008/export-to-xls-xlsx

Chapter 5: Fill report

Parameters

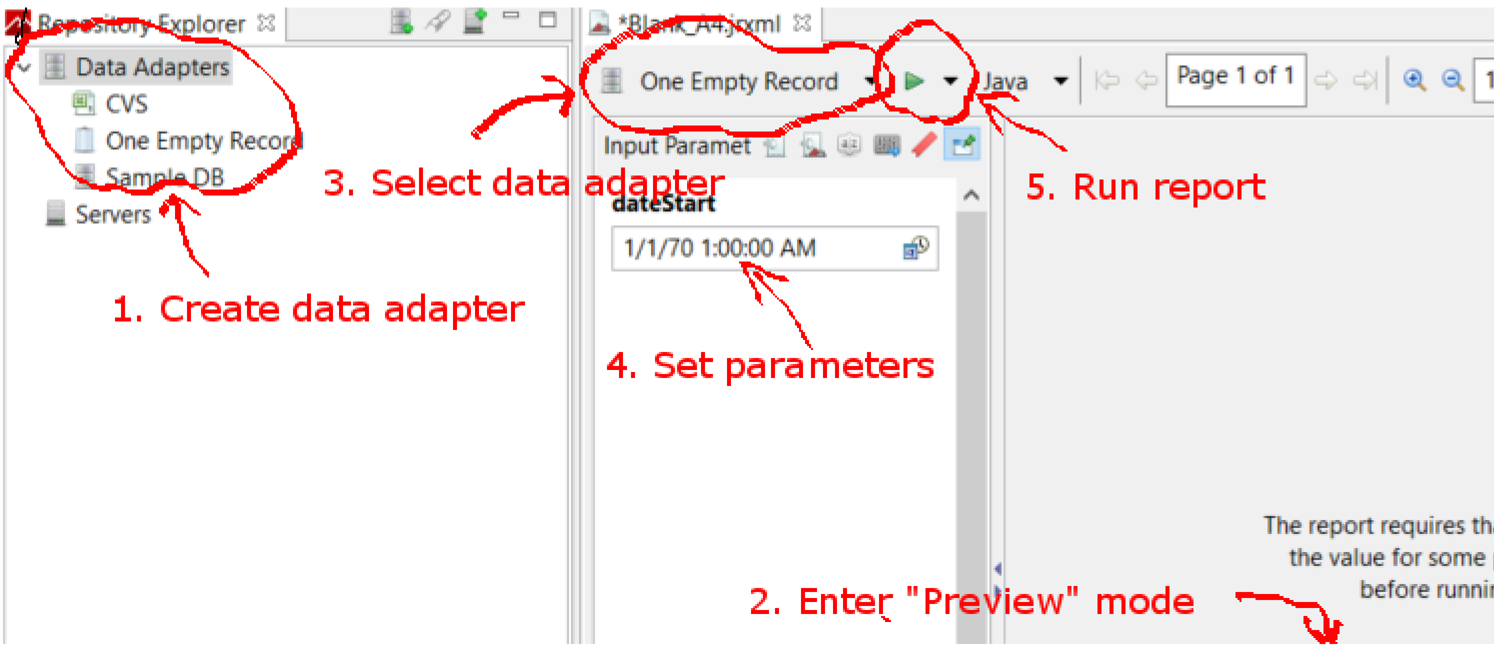
Parameters	Column
jasperPrint	The output of the fill process that can be exported to desired format
reportTemplate	The compiled design file `.jasper`
parameters	The parameter Map, that if defined can be references inside report by `$P{key}`
datasource	A net.sf.jasperreports.engine.JRDataSource
connection	A database connection java.sql.Connection

Examples

With IDE (Integrated development environment)

JasperSoft Studio

1. If datasource or database connection is needed to fill report, create your Data Adapter in Repository Explorer by right clicking "Data Adapters" selecting "Create Data Adapter"
2. Enter preview mode by selecting the **Preview** tab (no errors in deign need to be present)
3. Select desired dastasource (if no datasource is required select "One Empty Record"
4. Set parameter as desired
5. Fill report by clicking the green arrow "Run the report"



Fill JasperReport Template using Java

Common Requirements

All reports, regardless of how the data is presented, take a path to the report template and a parameter map. The variables are used in all examples that follow:

```
// 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";
```

Using a `.jrxml` file incurs an extra compilation step that isn't necessary in most situations. Unless you've written custom software to change the `.jrxml` before the report runs (e.g., adding or removing columns dynamically), use the `.jasper` file as shown in the subsequent examples.

Using a Database Connection

```
// 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);
```

Using a Custom Data Source

```
// 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);
```

Without Data Source, unused Detail Band

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

> Without a datas ource, the attribute `whenNoDataType="AllSectionsNoDetail"` on the `JasperReport` element must be set, otherwise an empty (blank) report will be generated.

Read Fill report online: https://riptutorial.com/jasper-reports/topic/3958/fill-report

Chapter 6: Font-extensions

Examples

Creating and using font extensions

Create a font extension using the IDE. See the iReport or Jaspersoft Studio documentation for details. The font extension can also be created manually.

What are font extensions?

Using a `textElement` you can specify a font (if not specified default font `SansSerif` is used)

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

To calculate font-metric (for line breaks, alignment etc) and render the font correctly, the **font** needs to be **mapped in the JVM** (Java virtual macchine). You could install the font file directly to the JVM but this is not encourage

From the JasperReport Ultimate Guide:

> We strongly encourage people to use only fonts derived from font extensions, because this is the only way to make sure that the fonts will be available to the application when the reports are executed at runtime. Using system fonts always brings the risk for the reports not to work properly when deployed on a new machine that might not have those fonts installed

Default font extension

JasperReports provide a default font-extension (see maven distribution jasperreports-fonts.jar). Adding this to classpath you can use the following fontName's without creating your own font-extension

> DejaVu Sans
> DejaVu Serif
> DejaVu Sans Mono

Common Issues

Issues to consider when using font's in pdf (itext):

- When exporting to PDF, if the text is not rendered correctly (missing parts, characters not showed, not wrapping or sized correctly), the **font-extensions** are likely missing.
- Is the actual `.tff` **supported** (OpenType) and can the font actually **render** the character? Not all fonts render all characters in `UTF-8`.
- Is the **correct encoding** passed to iText? In doubts (or in general) use the **encoding** **`Identity-H`** this is recommend for newer PDF standards and gives you the ability to mix different encoding.
- Is the font **embedded** so that a PDF shared across computers can display the content even if the font is not installed? If the font is not one of the 14 Standard Type 1 fonts always embed it.

Note the version of iText used by jasper report will not render all fonts (ligaturizer problem), You can test the `ttf` font and encoding directly see How can I test if my font is rendered correctly in pdf?

Read Font-extensions online: https://riptutorial.com/jasper-reports/topic/5773/font-extensions

Chapter 7: Using subreports

Parameters

Parameter	Details
parametersMapExpression	The Map with parameters. *Not required*
subreportParameter	The pair of name and value (set with ***subreportParameterExpression***). *Not required*. Several parameters can be passed to subreport
connectionExpression	Connection for getting data. *Not required*
dataSourceExpression	Expression for passing Datasource. *Not required*
subreportExpression	The subreport's path/URI or even JasperReport object. *Not required*
returnValue	The pair of name and value. *Not required*. Several values can be returned from subreport to master report back

Remarks

- Subreports can be used for constructing complex reports. The reusing of existing reports is another goal of using subreports.
- The subreport will be shown as a part of master report in case using of `<subreport>` element.
- The value of ***subreportExpression*** parameter is differ for using at *JasperReports Server* or just by *JasperReports* framework (some *API* using or using in IDE).

  For *JasperReports Server* it looks like:

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

  For using by just *JasperReports* engine:

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

  The great explanation by *@AndreasDietrich* can be found at *JasperServer: Unable to locate the subreport exception* post

- For some reasons the subreport can be used as a common report - without calling from the master report (with help of `<subreport>` element). The subreport is always a report.

Examples

Passing connection to subreport; return values back to the master report

This is a snippet of master report. Two parameters and the connection (for example, *jdbc*) are passing to the subreport. One value is returned from the subreport back to the master report, this value (*variable*) can be used in master report

```
<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>
```

Passing datasoure to subreport

This is a snippet of master report. The datasource is passed to the subreport with help of ***net.sf.jasperreports.engine.data.JRBeanCollectionDataSource*** constructor

```
<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>
```

Read Using subreports online: https://riptutorial.com/jasper-reports/topic/5452/using-subreports

Credits

S. No	Chapters	Contributors
1	Getting started with jasper-reports	Alex K, Community, Dave Jarvis, Petter Friberg
2	Compile JasperReports .jrxml to .jasper	Alex K, Dave Jarvis, Petter Friberg
3	Export to pdf	Alex K, Dave Jarvis, Petter Friberg, RamenChef
4	Export to xls/xlsx	Alex K
5	Fill report	Alex K, Dave Jarvis, Petter Friberg
6	Font-extensions	Dave Jarvis, Petter Friberg
7	Using subreports	Alex K