EBook Gratuito

APPRENDIMENTO jasper-reports

Free unaffiliated eBook created from
Stack Overflow contributors.

#jasper-reports

Sommario

Di

Capitolo 1: Inizia con jasper-reports

Osservazioni

Ci sono diverse librerie usate *API JasperReports Java* per creare report con Java:

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

Queste librerie / framework possono creare report "on-fly" con o senza l'uso del template del report (file *jrxml*)

Versioni

Libreria *JasperReports*

Versione	Data di rilascio
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

Versione	Data di rilascio
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 per la progettazione di report

La versione corrente del designer è basata su *Eclipse* : *Jaspersoft Studio* .

La versione precedente del designer era basata su *NetBeans* : *iReport Desigher* .

La prima versione di *iReport Designer* era un'applicazione indipendente - *iReport Classic*

Examples

Installazione o configurazione

Libreria JasperReports

JasperReports è uno strumento di reporting basato su Java open source. La libreria *JasperReports* può essere scaricata dalla comunità Jaspersoft per l'ultima versione .

> Nelle versioni recenti i jar di terze parti nella cartella lib **non** sono distribuiti, devono essere scaricati da repository pubblici, vedere `pom.xml` distribuito per le dipendenze. Maven può essere usato per recuperare tutte le dipendenze comprese quelle transitorie nella cartella target / dipendenza.

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

Jaspersoft Studio (IDE)

Jaspersoft Studio è il client di progettazione ufficiale per JasperReports, basato sulla piattaforma Eclipse, per sostituire iReport Designer.

iReport Designer (IDE)

iReport Designer è il precedente progettista di report per JasperReports. La versione 5.6.0 (rilasciata a maggio 2014) era l'ultima versione ufficiale; il supporto del fornitore è terminato alla fine del 2015.

Risorse JasperReport Commuity

Domande frequenti sulla libreria JasperReports

- FAQ

Codice sorgente

- JasperReports Codice sorgente della libreria

Esercitazioni

- Punto tutorial

- JasperReports Guida definitiva

Campioni

- Esempio di riferimento

Riferimenti

- Documentazione ufficiale
- Wiki della comunità

Bug Tracker ufficiale

- Bug Tracker

Flusso di lavoro

Il flusso di lavoro in jasper-report è:

1. Progettare il report, creare il file jrxml che definisce il layout del report. Il jrxml può essere creato usando un semplice texteditor ma normalmente un IDE (JasperSoft Studio o iReport) viene utilizzato sia per velocizzare lo sviluppo del report, ma anche per avere una visuale del layout.

2. Compilare il report (il jrxml) per ottenere un file .jasper o un oggetto JasperReport . Questo processo può essere confrontato con un file `.java` compilato in `.class` .

3. Compilare il report , passare i parametri e un'origine dati al report per generare l'oggetto di stampa JasperPrint che può anche essere salvato in un file `.jprint`

4. Visualizza, stampa e / o esporta il JasperPrint. Il formato di esportazione più comune è supportato come pdf, excel, word, html, cvs ecc.

Comprensione delle diverse bande di report

Titolo

Questa band viene mostrata una volta all'inizio del report. Può essere usato come prima pagina impostando l'attributo `isTitleNewPage="true"`

Intestazione di pagina

Questo appare all'inizio di ogni pagina esclusa la prima pagina se viene utilizzata la banda del titolo e l'ultima pagina se la banda di riepilogo viene utilizzata con l'impostazione `isSummaryWithPageHeaderAndFooter="false"`

Intestazione colonna

Questo appare prima della banda di dettaglio in ogni pagina.

Dettaglio

Questa sezione è iterata **per ogni record** nell'origine dati fornita. È consentito avere più banda di dettaglio (dettaglio 1, dettaglio 2 .. dettaglio n), vengono iterati come segue

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

Colonna Piè di pagina

Questo appare sotto la banda di dettaglio in ogni pagina in cui è presente la banda di dettaglio. L'impostazione predefinita è fine pagina (prima del piè di pagina), ma può essere `isFloatColumnFooter="true"` sotto l'ultima banda di dettagli (ultimo record) impostando l'attributo `isFloatColumnFooter="true"`

Piè di pagina

Questo appare nella parte inferiore di ogni pagina, escludendo la banda del titolo, la banda di riepilogo (senza il piè di pagina) e l'ultima banda non di riepilogo se viene utilizzato il piè di pagina dell'ultima pagina.

Ultimo piè di pagina

Questo appare nell'ultima pagina (se non nella banda di riepilogo senza piè di pagina) invece del normale piè di pagina

Sommario

Questo appare alla fine del report nella nuova pagina se `isSummaryNewPage="true"` è impostato e con l'intestazione e il piè di pagina se `isSummaryWithPageHeaderAndFooter="true"`

Intestazione di gruppo

Questa sezione appare se un gruppo viene definito ogni volta che cambia l'espressione del gruppo, prima della banda di dettaglio.

Piè di pagina di gruppo

Questa sezione appare se un gruppo viene definito ogni volta *prima che* l'espressione del gruppo cambi, dopo la banda di dettaglio.

sfondo

Questa banda viene visualizzata su ogni pagina come sfondo di tutte le altre bande.

Nessun dato

Appare solo se non è stata passata alcuna origine dati o se l'origine dati è vuota (0 record) e `whenNoDataType="NoDataSection"` è impostato.

Jasper riporta i formati di file

- `.jrxml` è il file di progettazione del report, il suo formato è in XML leggibile dall'uomo, può essere soddisfatto in un oggetto `JasperReport` e salvato come `.jasper`
- `.jasper` è la versione compilata di `.jrxml` e può essere caricata direttamente in un oggetto `JasperReport` pronto per essere riempito con i dati
- `.jrprint` è l'oggetto `JasperPrint` serializzato, un report che è già stato riempito con i dati e può essere caricato per essere stampato, visualizzato e / o esportato nel formato desiderato.
- `.jrpxml` è il rappresentativo XML di un oggetto `JasperPrint` che può essere modificato e quindi unmarsarizzato per recuperare l'oggetto `JasperPrint`

Leggi Inizia con jasper-reports online: https://riptutorial.com/it/jasper-reports/topic/3594/inizia-con-jasper-reports

Capitolo 2: Compila il rapporto

Parametri

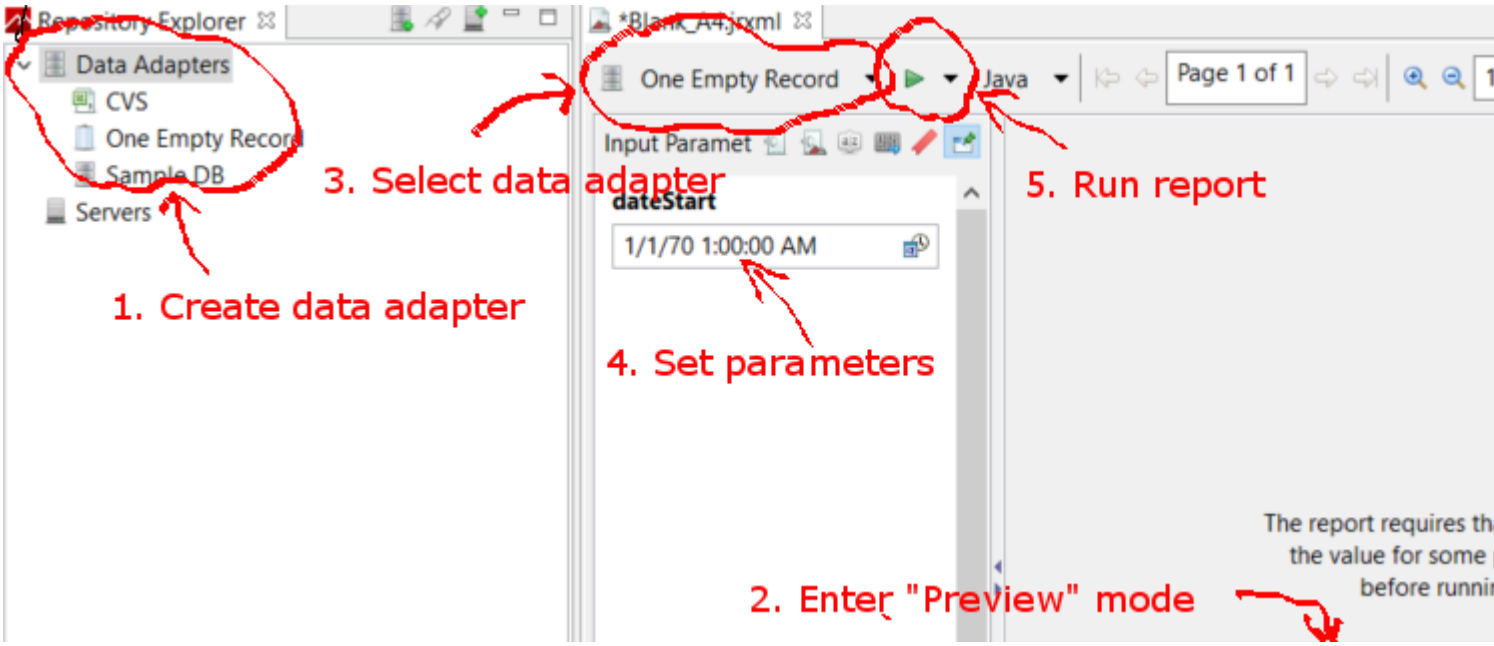
parametri	Colonna
JasperPrint	L'output del processo di riempimento che può essere esportato nel formato desiderato
reportTemplate	Il file di disegno compilato `.jasper`
parametri	Il parametro Map , che se definito può essere riferimento all'interno del report di `$P{key}`
fonte di dati	A net.sf.jasperreports.engine.JRDataSource
connessione	Una connessione al database java.sql.Connection

Examples

Con IDE (ambiente di sviluppo integrato)

JasperSoft Studio

1. Se per compilare il report è necessaria la connessione al database o all'origine dati, creare la scheda dati in Repository Explorer facendo clic con il pulsante destro del mouse su "Schede dati" selezionando "Crea adattatore dati".
2. Entra nella modalità di anteprima selezionando la scheda **Anteprima** (non è necessario che siano presenti errori nel degno)
3. Selezionare l'origine dati desiderata (se non è richiesta l'origine dati selezionare "One Empty Record"
4. Impostare i parametri come desiderato
5. Compila il rapporto facendo clic sulla freccia verde "Esegui il rapporto"



Riempi il template JasperReport usando Java

Requisiti comuni

Tutti i report, indipendentemente da come vengono presentati i dati, seguono un percorso per il modello di report e una mappa dei parametri. Le variabili sono utilizzate in tutti gli esempi che seguono:

```
// 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'utilizzo di un file `.jrxml` comporta un passaggio di compilazione aggiuntivo che non è necessario nella maggior parte delle situazioni. A meno che tu non abbia scritto un software personalizzato per modificare il file `.jrxml` prima `.jrxml` del rapporto (ad esempio, aggiungendo o rimuovendo le colonne in modo dinamico), utilizza il file `.jasper` come mostrato negli esempi successivi.

Utilizzo di una connessione al database

```
// 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);
```

Utilizzo di un'origine dati personalizzata

```
// 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);
```

Senza origine dati, banda dettagli inutilizzata

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

> Senza un dato, l'attributo `whenNoDataType="AllSectionsNoDetail"` sull'elemento `JasperReport` deve essere impostato, altrimenti verrà generato un report vuoto (vuoto).

Leggi Compila il rapporto online: https://riptutorial.com/it/jasper-reports/topic/3958/compila-il-rapporto

Capitolo 3: Compilare JasperReports .jrxml a .jasper

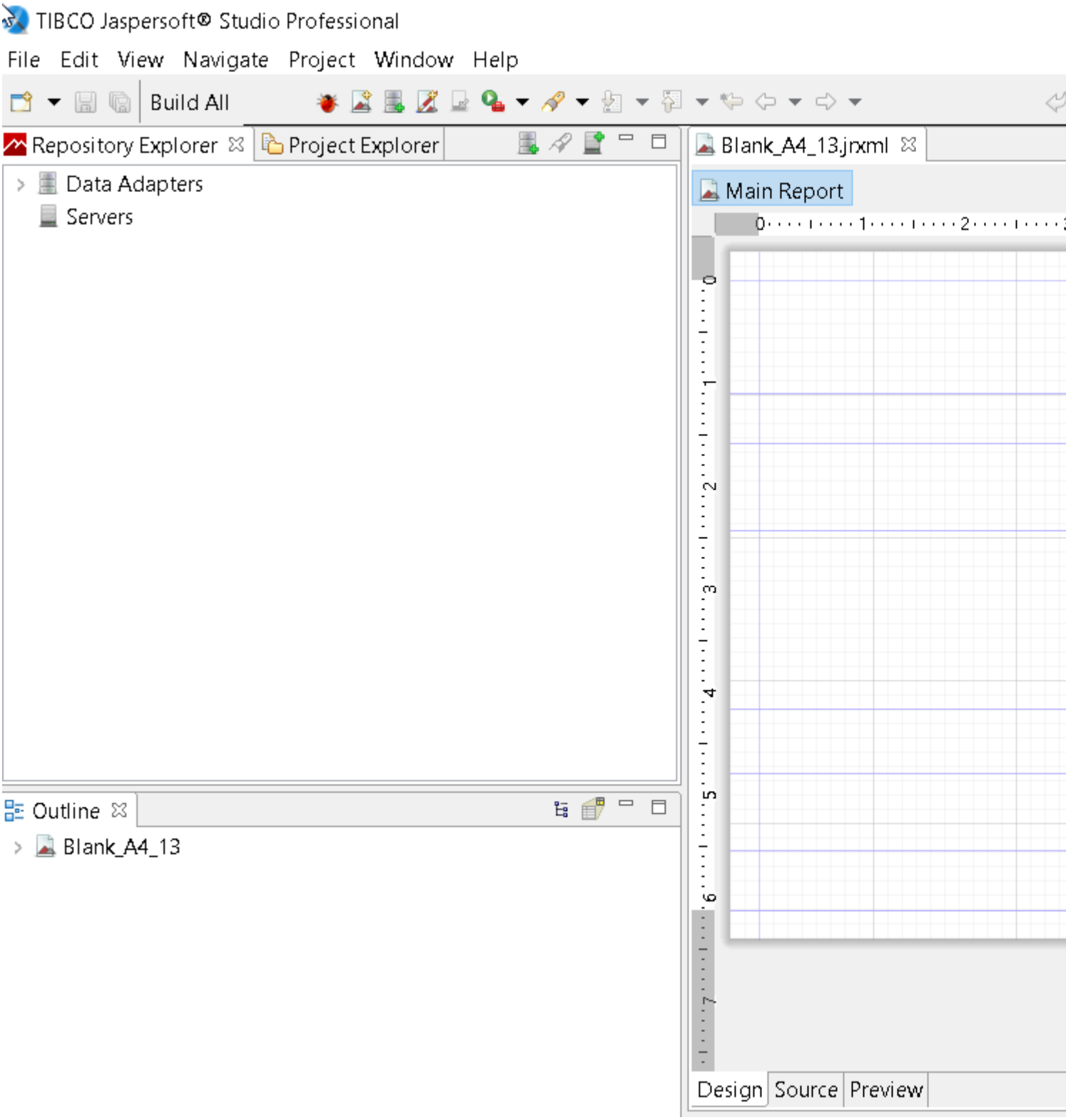
Examples

Con IDE (ambiente di sviluppo integrato)

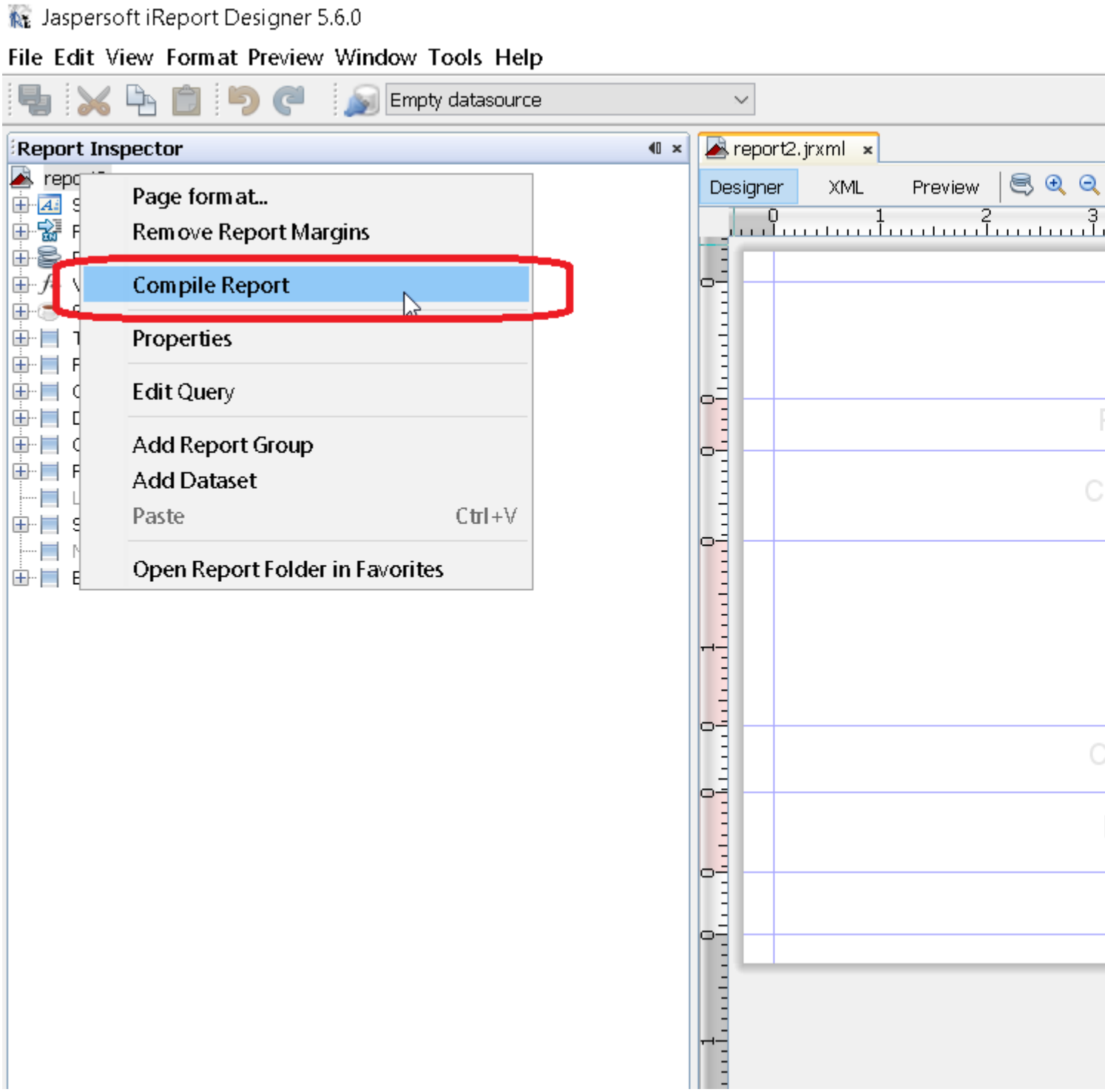
In IDE Jaspersoft Studio (*JSS*) o nella versione precedente iReport Designer è sufficiente premere **Anteprima** .

Il file di disegno JasperReports `.jrxml` verrà automaticamente compilato in `.jasper` nella stessa cartella di `.jrxml` se **non** sono presenti **errori** .

Un altro modo è quello di premere il pulsante *"Compila rapporto"* in *JSS*



oppure utilizzare il menu di scelta rapida *"Compile Report"* chiamato da *Report Inspector* in *iReport*



Con 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>
```

Lo strumento di compilazione di Apache Ant deve essere installato correttamente sul tuo sistema

Con Java

Mentre è possibile compilare i file `.jrxml` in file `.jasper` usando il codice Java, questo comporta un `.jasper` prestazioni che è meglio evitare pre-compilando i file `.jrxml` usando l'IDE. Con questo in mente, la compilazione di file `.jrxml` può essere eseguita utilizzando JasperCompileManager come segue:

```
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
```

Con Apache Maven

Il *plugin JasperReports* di Alex Nederlof è una buona alternativa di org.codehaus.mojo abbandonato : jasperreports-maven-plugin plugin.

L'aggiunta di plug-in è una procedura semplice e tipica:

```
<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>
```

Il comando per la compilazione con *Maven* :

mvn jasperreports: jasper

I file *jasper* verranno creati nella cartella *$ {project.build.directory} / jasper* (ad esempio, in */ target / jasper*)

Leggi Compilare JasperReports .jrxml a .jasper online: https://riptutorial.com/it/jasper-reports/topic/4943/compilare-jasperreports--jrxml-a--jasper

Capitolo 4: Esporta in pdf

Osservazioni

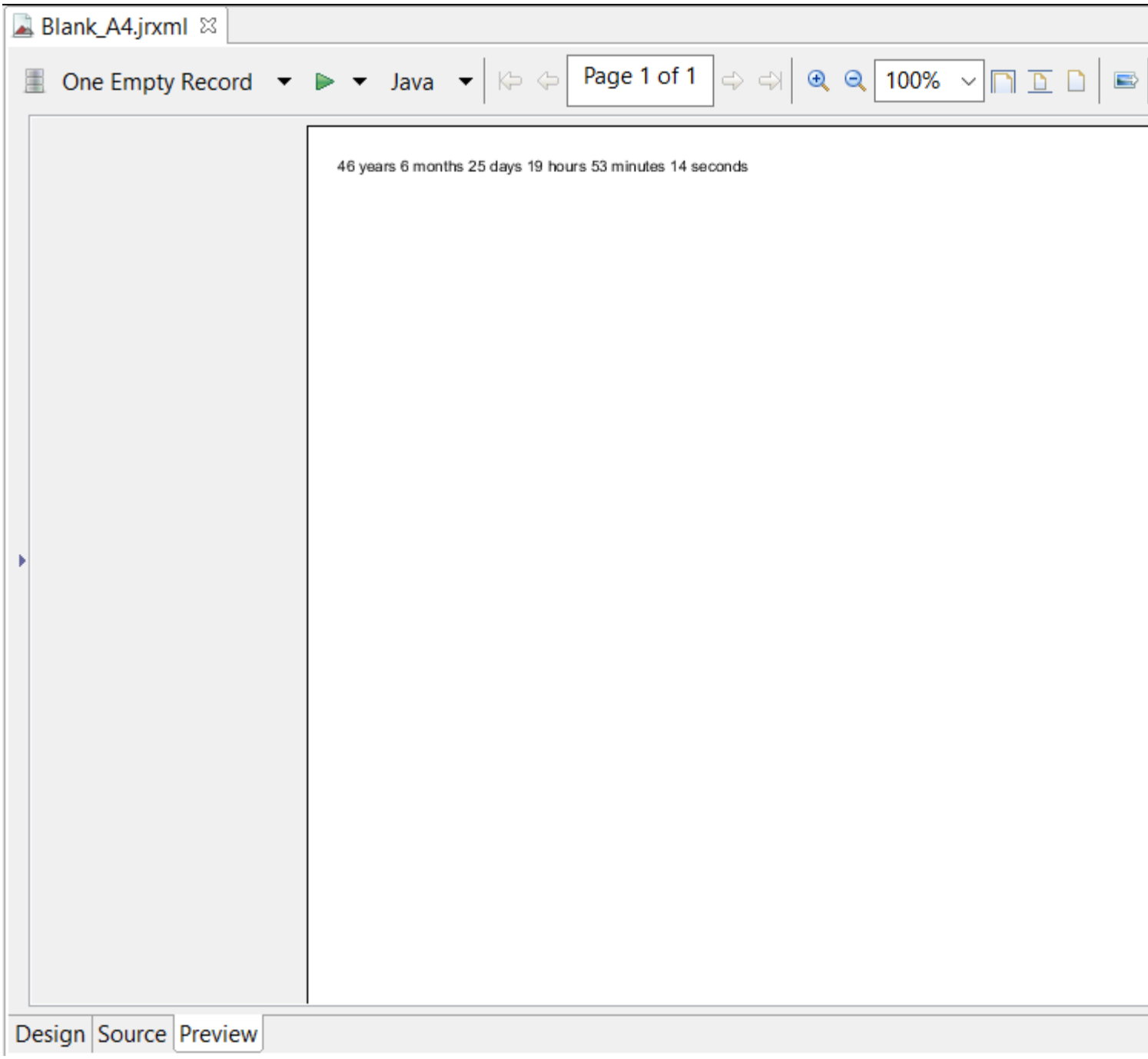
Per rendere i **font** corretti in font-extension pdf dovrebbe sempre essere usato (in classpath)

Examples

Con IDE (ambiente di sviluppo integrato)

JasperSoft Studio

In Anteprima, esegui il rapporto facendo clic sulla freccia verde, se non ci sono errori il menu di esportazione sarà abilitato, fai clic sul pulsante di esportazione (immagine del disco) e seleziona "Esporta come Pdf"



Con Java

Per esportare a è necessario compilare il report per ottenere l'oggetto JasperPrint .

Esportare singoli file JasperPrint (single jrxml) in un 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();
```

Esportare più JasperPrint (più jrxml) in un singolo file

Solo i primi passi differiscono dal set precedente:

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

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

I passaggi rimanenti sono gli stessi:

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

Vedi API SimplePdfExporterConfiguration per i dettagli di configurazione.

Leggi Esporta in pdf online: https://riptutorial.com/it/jasper-reports/topic/4190/esporta-in-pdf

Capitolo 5: Esporta in xls / xlsx

Examples

Con Java

Esporta in formato *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();
}
```

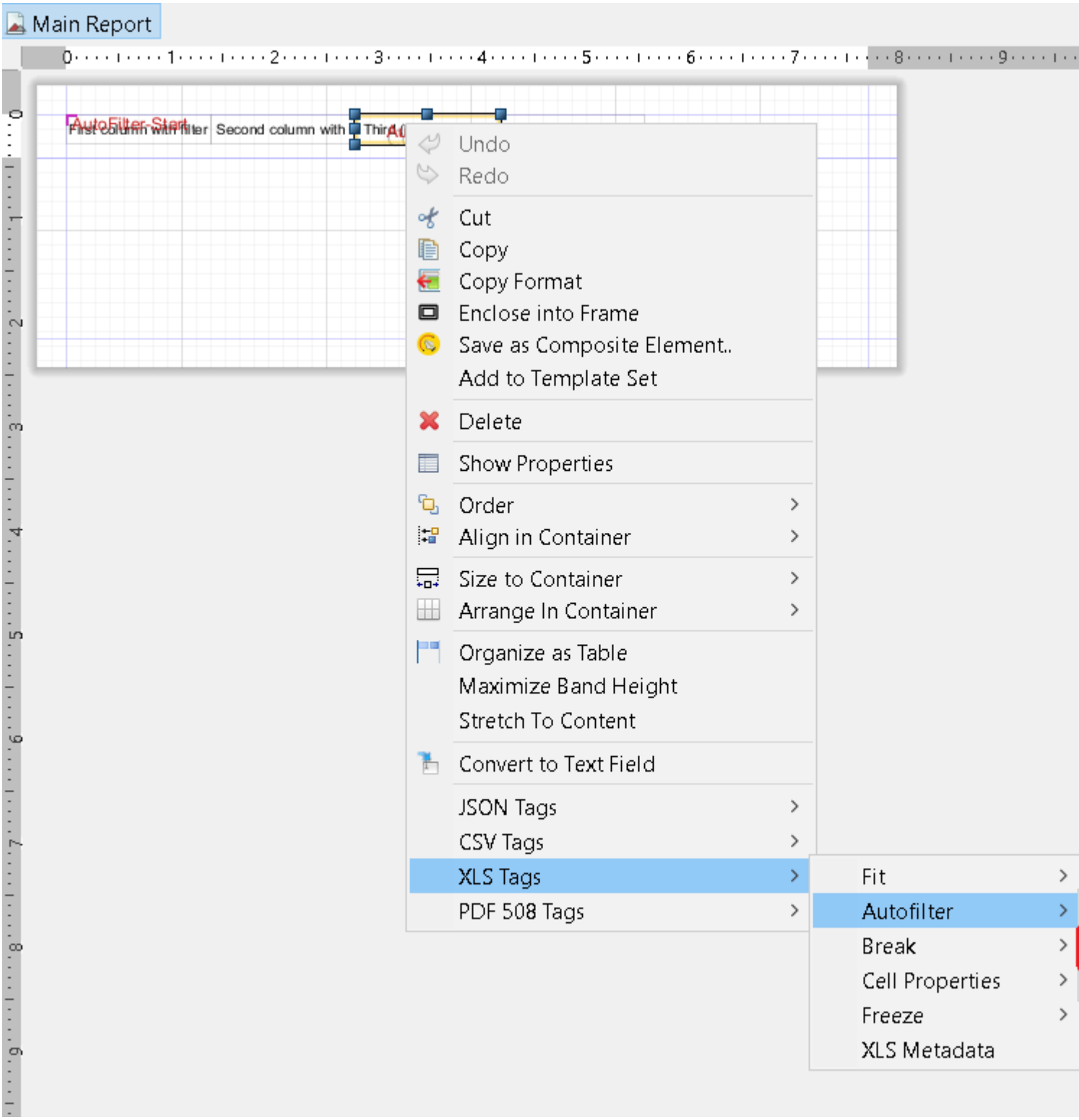
Aggiunta del filtro automatico per le colonne

L'utilizzo della proprietà ***net.sf.jasperreports.export.xls.auto.filter*** consente di aggiungere il filtro automatico nel file xls generato.

```
<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 proprietà può essere impostata in *Jaspersoft Studio* con l'aiuto del menu di scelta rapida o manualmente modificando il file *jrxml* .



Leggi Esporta in xls / xlsx online: https://riptutorial.com/it/jasper-reports/topic/5008/esporta-in-xls---

xlsx

Capitolo 6: Font-extensions

Examples

Creazione e utilizzo delle estensioni dei caratteri

Crea un'estensione di carattere usando l'IDE. Consultare la documentazione di iReport o Jaspersoft Studio per i dettagli. L'estensione del carattere può anche essere creata manualmente .

Quali sono le estensioni dei caratteri?

Usando un `textElement` puoi specificare un font (se non viene specificato il carattere predefinito `SansSerif`)

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

Per calcolare la metrica dei caratteri (per interruzioni di linea, allineamento, ecc.) E renderizzare correttamente il **carattere** , il **carattere** deve essere **mappato nella JVM** (Java virtual macchine). È possibile installare il file di font direttamente sulla JVM, ma questo non è un incoraggiamento

Dalla guida definitiva di JasperReport:

> Incoraggiamo vivamente le persone a utilizzare solo caratteri derivati da estensioni di caratteri, poiché questo è l'unico modo per assicurarsi che i caratteri siano disponibili per l'applicazione quando i report vengono eseguiti in fase di runtime. L'uso dei caratteri di sistema comporta sempre il rischio che i report non funzionino correttamente quando vengono distribuiti su una nuova macchina che potrebbe non avere quei font installati

Estensione di carattere predefinita

JasperReports fornisce un'estensione carattere predefinita (vedi distribuzione maven jasperreports-fonts.jar). Aggiungendo questo a classpath puoi usare i seguenti fontName senza creare la tua estensione font

> DejaVu Sans
> DejaVu Serif
> DejaVu Sans Mono

Problemi comuni

Problemi da considerare quando si usano i font in pdf (itext):

- Quando si esporta in PDF, se il testo non viene reso correttamente (parti mancanti, caratteri non mostrati, non avvolti o dimensionati correttamente), le **estensioni** dei **caratteri** sono probabilmente mancanti.
- L'attuale `.tff` **supportato** (OpenType) e il font può effettivamente **renderizzare** il personaggio? Non tutti i font rendono tutti i caratteri in `UTF-8` .
- La **codifica corretta è** passata a iText? In caso di dubbi (o in generale) utilizzare la **codifica** `Identity-H` è consigliabile per gli standard PDF più recenti e offre la possibilità di combinare diverse codifiche.
- Il carattere è **incorporato in** modo che un PDF condiviso tra i computer possa visualizzare il contenuto anche se il carattere non è installato? Se il carattere non è uno dei 14 caratteri standard di tipo 1, incorpora sempre questo carattere.

Nota che la versione di iText usata da jasper report non renderà tutti i font (problema di ligaturizer), puoi testare il font `ttf` e la codifica direttamente vedi Come posso testare se il mio font è reso correttamente in pdf?

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

Capitolo 7: Utilizzo dei sottoreport

Parametri

Parametro	Dettagli
parametersMapExpression	La mappa con parametri. *Non richiesto*
subreportParameter	La coppia di nome e valore (impostata con ***subreportParameterExpression***). *Non richiesto* Diversi parametri possono essere passati al sottoreport
connectionExpression	Connessione per ottenere dati. *Non richiesto*
dataSourceExpression	Espressione per il passaggio di Datasource. *Non richiesto*
subreportExpression	Percorso / URI del sottoreport o anche oggetto JasperReport. *Non richiesto*
valore di ritorno	La coppia di nome e valore. *Non richiesto* Diversi valori possono essere restituiti dal sottoreport al report master indietro

Osservazioni

- I sottoreport possono essere utilizzati per la creazione di report complessi. Il riutilizzo dei report esistenti è un altro obiettivo dell'utilizzo dei sottoreport.
- Il sottoreport verrà visualizzato come parte del report principale in caso di utilizzo dell'elemento `<subreport>` .
- Il valore del parametro ***subreportExpression*** è diverso per l'utilizzo su *JasperReports Server* o semplicemente sul framework *JasperReports* (alcune *API che* utilizzano o utilizzano in IDE).

  Per *JasperReports Server* sembra che:

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

  Per l'utilizzo con il solo motore *JasperReports* :

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

  La grande spiegazione di *@AndreasDietrich* può essere trovata su *JasperServer: impossibile individuare il* messaggio *dell'eccezione del sottoreport*

- Per alcuni motivi, il sottoreport può essere utilizzato come report comune, senza chiamare dal report principale (con l'aiuto dell'elemento `<subreport>`). Il sottoreport è sempre un report.

Examples

Passare la connessione al sottoreport; restituire i valori al report principale

Questo è uno snippet di report principale. Due parametri e la connessione (ad esempio, *jdbc*) passano al sottoreport. Un valore viene restituito dal sottoreport al report principale, questo valore (*variabile*) può essere utilizzato nel report principale

```
<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>
```

Passare datasoure a sottoreport

Questo è uno snippet di report principale. L'origine dati viene passata al sottoreport con l'aiuto di ***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>
```

Leggi Utilizzo dei sottoreport online: https://riptutorial.com/it/jasper-reports/topic/5452/utilizzo-dei-sottoreport

Titoli di coda

S. No	Capitoli	Contributors
1	Inizia con jasper-reports	Alex K, Community, Dave Jarvis, Petter Friberg
2	Compila il rapporto	Alex K, Dave Jarvis, Petter Friberg
3	Compilare JasperReports .jrxml a .jasper	Alex K, Dave Jarvis, Petter Friberg
4	Esporta in pdf	Alex K, Dave Jarvis, Petter Friberg, RamenChef
5	Esporta in xls / xlsx	Alex K
6	Font-extensions	Dave Jarvis, Petter Friberg
7	Utilizzo dei sottoreport	Alex K