EBook Gratis

APRENDIZAJE jasper-reports

Free unaffiliated eBook created from
Stack Overflow contributors.

#jasper-reports

Tabla de contenido

Acerca de ... 1

Capítulo 1: Comenzando con los informes de jaspe ... 2

Observaciones ... 2

Versiones ... 2

Biblioteca JasperReports ... 2

IDE para el diseño de informes ... 3

Examples ... 4

Instalación o configuración ... 4

JasperReports Library ... 4

Jaspersoft Studio (IDE) ... 4

iReport Designer (IDE) ... 4

Recursos de JasperReport Commuity ... 4

Preguntas frecuentes de JasperReports Library ... 4

Código fuente ... 4

Tutoriales ... 4

Muestras ... 5

Referencias ... 5

Bug Tracker oficial ... 5

Flujo de trabajo ... 5

Entendiendo las diferentes bandas de informes ... 5

Título ... 5

Encabezado de página ... 5

Encabezado de la columna ... 5

Detalle ... 6

Pie de columna ... 6

Pie de página ... 6

Último pie de página ... 6

Resumen ... 6

Encabezado de grupo ... 6

Pie de grupo ... 7
Fondo ... 7
Sin datos ... 7
Formatos de archivo de informe Jasper ... 7
Capítulo 2: Compile JasperReports .jrxml a .jasper ... 8
Examples ... 8
Con IDE (entorno de desarrollo integrado) ... 8
Con Apache Ant ... 10
Con java ... 11
Con apache maven ... 11
Capítulo 3: Exportar a pdf ... 13
Observaciones ... 13
Examples ... 13
Con IDE (entorno de desarrollo integrado) ... 13
JasperSoft Studio ... 13
Con java ... 14
Exportar JasperPrint único (solo jrxml) al archivo ... 14
Exportar múltiples JasperPrint's (multiple jrxml) a un solo archivo ... 15
Capítulo 4: Exportar a xls / xlsx ... 16
Examples ... 16
Con java ... 16
Añadiendo autofiltro para columnas ... 16
Capítulo 5: Extensiones de fuente ... 19
Examples ... 19
Creación y uso de extensiones de fuente ... 19
¿Qué son las extensiones de fuente? ... 19
Extensión de fuente predeterminada ... 19
Problemas comunes ... 19
Capítulo 6: Llenar informe ... 21
Parámetros ... 21
Examples ... 21

Con IDE (entorno de desarrollo integrado) 21
JasperSoft Studio 21
Rellene la plantilla de JasperReport usando Java 22
Requisitos comunes 22
Usando una conexión de base de datos 22
Usando una fuente de datos personalizada 22
Sin fuente de datos, banda de detalle no utilizada 23
Capítulo 7: Usando subinformes 24
Parámetros 24
Observaciones 24
Examples 25
Pasando la conexión al subinforme; devuelve los valores al informe maestro 25
Pasar datasoure a subinforme 25
Creditos 26

Acerca de

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

Capítulo 1: Comenzando con los informes de jaspe

Observaciones

Hay varias bibliotecas que utilizan *la API de Java de JasperReports* para crear informes con Java:

- **Informes dinámicos**
- **DynamicJasper**

Estas bibliotecas / marcos pueden generar informes "al vuelo" con o sin usar la plantilla del informe (archivo *jrxml*)

Versiones

Biblioteca *JasperReports*

Versión	Fecha de lanzamiento
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

Versión	Fecha de lanzamiento
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 para el diseño de informes.

La versión actual del diseñador se basa en *Eclipse* : *Jaspersoft Studio* .

La versión anterior del diseñador estaba basada en *NetBeans* : *iReport Desigher* .

La primera versión de *iReport Designer* fue una aplicación independiente - *iReport Classic*

Examples

Instalación o configuración

JasperReports Library

JasperReports es una herramienta de informes basada en Java de código abierto. La Biblioteca de *JasperReports* se puede descargar de la Comunidad de Jaspersoft para el último lanzamiento .

> En versiones recientes, los archivos `pom.xml` de terceros en la carpeta lib **no se** distribuyen, deben descargarse desde repositorios públicos, consulte `pom.xml` distribuido para conocer las dependencias. Se puede usar Maven para recuperar todas las dependencias, incluidas las transitorias en la carpeta de destino / dependencia.

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

Jaspersoft Studio (IDE)

Jaspersoft Studio es el cliente de diseño oficial de JasperReports, construido sobre la plataforma Eclipse, para reemplazar a iReport Designer.

iReport Designer (IDE)

iReport Designer es el diseñador de informes anterior para JasperReports. La versión 5.6.0 (lanzada en mayo de 2014) fue la última versión oficial; El soporte de proveedores finalizó a finales de 2015.

Recursos de JasperReport Commuity

Preguntas frecuentes de JasperReports Library

- Preguntas más frecuentes

Código fuente

- Código fuente de JasperReports Library

Tutoriales

- Punto de tutoriales
- JasperReports Ultimate Guide

Muestras

- Referencia de muestra

Referencias

- Documentacion oficial
- Comunidad Wiki

Bug Tracker oficial

- Localizador de bichos

Flujo de trabajo

El flujo de trabajo en jasper-reports es:

1. Diseñe el informe, cree el archivo jrxml que define el diseño del informe. El jrxml se puede crear utilizando un simple editor de texto, pero normalmente se usa un IDE (JasperSoft Studio o iReport) para acelerar el desarrollo del informe, pero también para tener una vista visual del diseño.

2. Compile el informe (el jrxml) para obtener un archivo .jasper o un objeto JasperReport . Este proceso puede ser comparado con un `.java` archivo que se está compilado para `.class` .

3. Rellene el informe , pase parámetros y una fuente de datos al informe para generar el objeto de impresión JasperPrint que también se puede guardar en un archivo `.jprint`

4. Ver, imprimir y / o exportar el JasperPrint. Los formatos de exportación más comunes son compatibles como pdf, excel, word, html, cvs, etc.

Entendiendo las diferentes bandas de informes.

Título

Esta banda se muestra una vez al principio del informe. Se puede usar como primera página configurando el atributo `isTitleNewPage="true"`

Encabezado de página

Esto aparece al principio de cada página, excluyendo la primera página si se usa la banda de título y la última página si se usa la banda de resumen con la configuración `isSummaryWithPageHeaderAndFooter="false"`

Encabezado de la columna

Esto aparece antes de la banda de detalle en cada página.

Detalle

Esta sección se itera **para cada registro** en la fuente de datos suministrada. Se permite tener múltiples bandas de detalle (detalle 1, detalle 2 ... detalle n), se repiten de la siguiente manera

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

Pie de columna

Esto aparece debajo de la banda de detalle en cada página donde está presente la banda de detalle. La configuración predeterminada es el final de la página (antes del pie de página), pero se puede cambiar a la última banda de detalle (último registro) estableciendo el atributo `isFloatColumnFooter="true"`

Pie de página

Aparece en la parte inferior de cada página, excluyendo la banda de título, la banda de resumen (sin pie de página) y la última banda sin resumen si se usa el Último pie de página.

Último pie de página

Esto aparece en la última página (si no es una banda de resumen sin pie de página) en lugar del pie de página normal

Resumen

Esto aparece al final del informe en la nueva página si `isSummaryNewPage="true"` está configurado y con el encabezado y el pie de página si `isSummaryWithPageHeaderAndFooter="true"`

Encabezado de grupo

Esta sección aparece si un grupo se define cada vez que cambia la expresión del grupo, antes de la banda de detalle.

Pie de grupo

Esta sección aparece si un grupo se define cada vez *antes* del cambio de expresión del grupo, después de la banda de detalle.

Fondo

Esta banda se muestra en cada página como fondo para todas las demás bandas.

Sin datos

Esto aparece solo si no se pasó ninguna fuente de datos o si la fuente de datos está vacía (0 registros) y `whenNoDataType="NoDataSection"` se establece `whenNoDataType="NoDataSection"` .

Formatos de archivo de informe Jasper

- `.jrxml` es el archivo de diseño de informe, su formato está en XML legible por humanos, se puede cumplir en un objeto `JasperReport` y guardarse como `.jasper`
- `.jasper` es la versión compilada del `.jrxml` y se puede cargar directamente en un objeto `JasperReport` listo para ser llenado con datos
- `.jrprint` es el objeto `JasperPrint` serializado, un informe que ya se ha llenado de datos y se puede cargar para imprimir, ver y / o exportar al formato deseado.
- `.jrpxml` es el rappresentativo XML de un objeto `JasperPrint` que se puede modificar y luego eliminar para recuperar el objeto `JasperPrint`

Lea Comenzando con los informes de jaspe en línea: https://riptutorial.com/es/jasper-reports/topic/3594/comenzando-con-los-informes-de-jaspe

Capítulo 2: Compile JasperReports .jrxml a .jasper

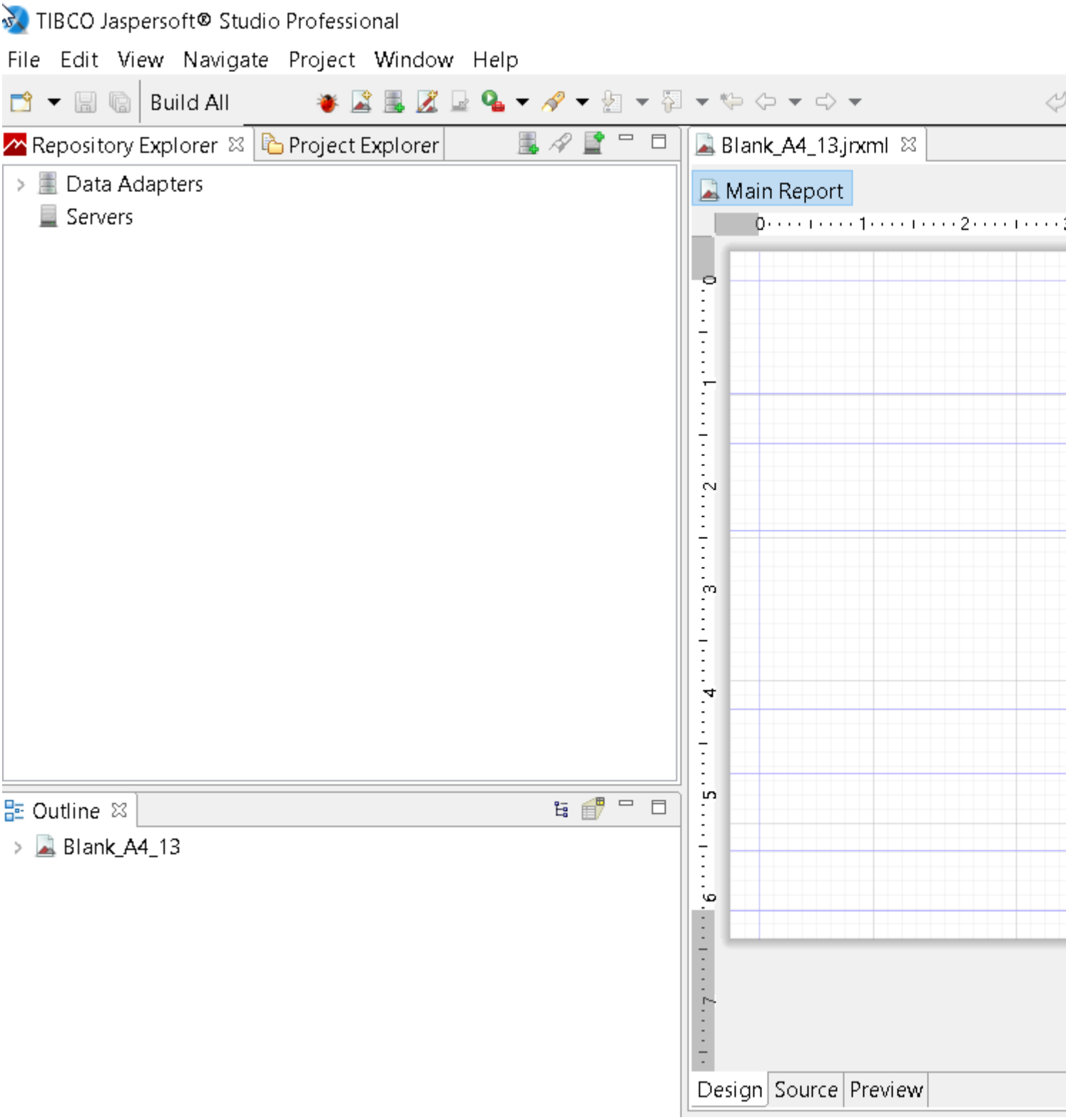
Examples

Con IDE (entorno de desarrollo integrado).

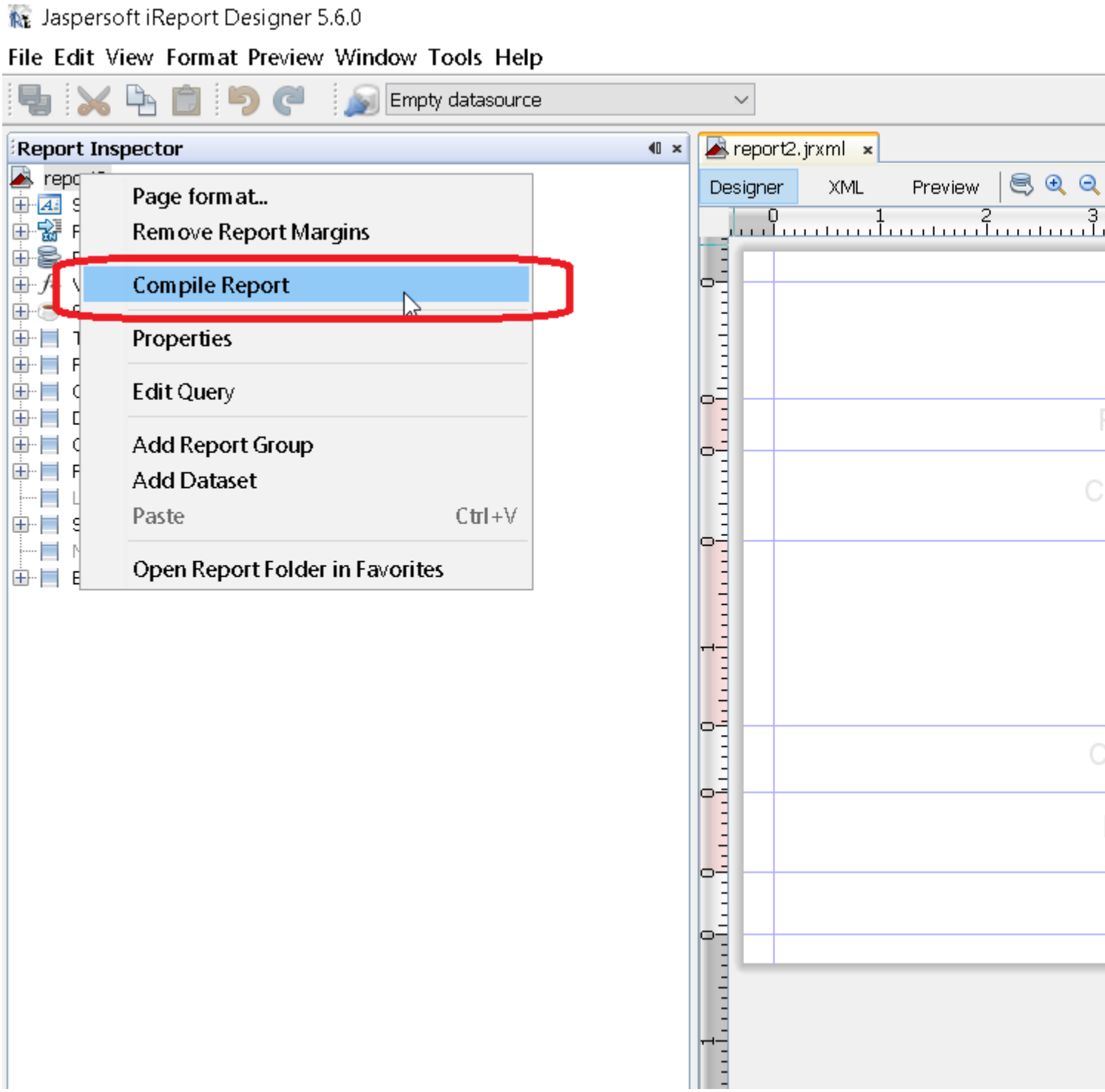
En IDE Jaspersoft Studio (*JSS*) o en la versión anterior de iReport Designer es suficiente presionar **Vista previa** .

El archivo de diseño de JasperReports `.jrxml` se compilará `.jasper` en `.jasper` en la misma carpeta que `.jrxml` si **no hay errores** presentes.

Otra forma es presionar el botón *"Compilar Informe"* en *JSS*



o use el menú contextual *"Compilar informe"* llamado desde el *Inspector de informes* en *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>
```

La herramienta de compilación Apache Ant debe estar correctamente instalada en su sistema

Con java

Si bien es posible compilar archivos `.jrxml` en archivos `.jasper` utilizando el código Java, esto genera un impacto en el rendimiento que se evita mejor compilando `.jrxml` archivos `.jrxml` utilizando el IDE. Teniendo esto en cuenta, la compilación de archivos `.jrxml` se puede realizar utilizando el JasperCompileManager de la siguiente manera:

```
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

El *JasperReports-plugin* de Alex Nederlof es una buena alternativa de org.codehaus.mojo abandonado : jasperreports-maven-plugin plugin.

La adición de plugin es un procedimiento típico y simple:

```
<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>
```

El comando para la compilación con *Maven* :

mvn jasperreports: jasper

Los archivos *jasper* se crearán en la carpeta *$ {project.build.directory}* / *jasper* (por ejemplo, en / *target* / *jasper*)

Lea Compile JasperReports .jrxml a .jasper en línea: https://riptutorial.com/es/jasper-reports/topic/4943/compile-jasperreports--jrxml-a--jasper

Capítulo 3: Exportar a pdf

Observaciones

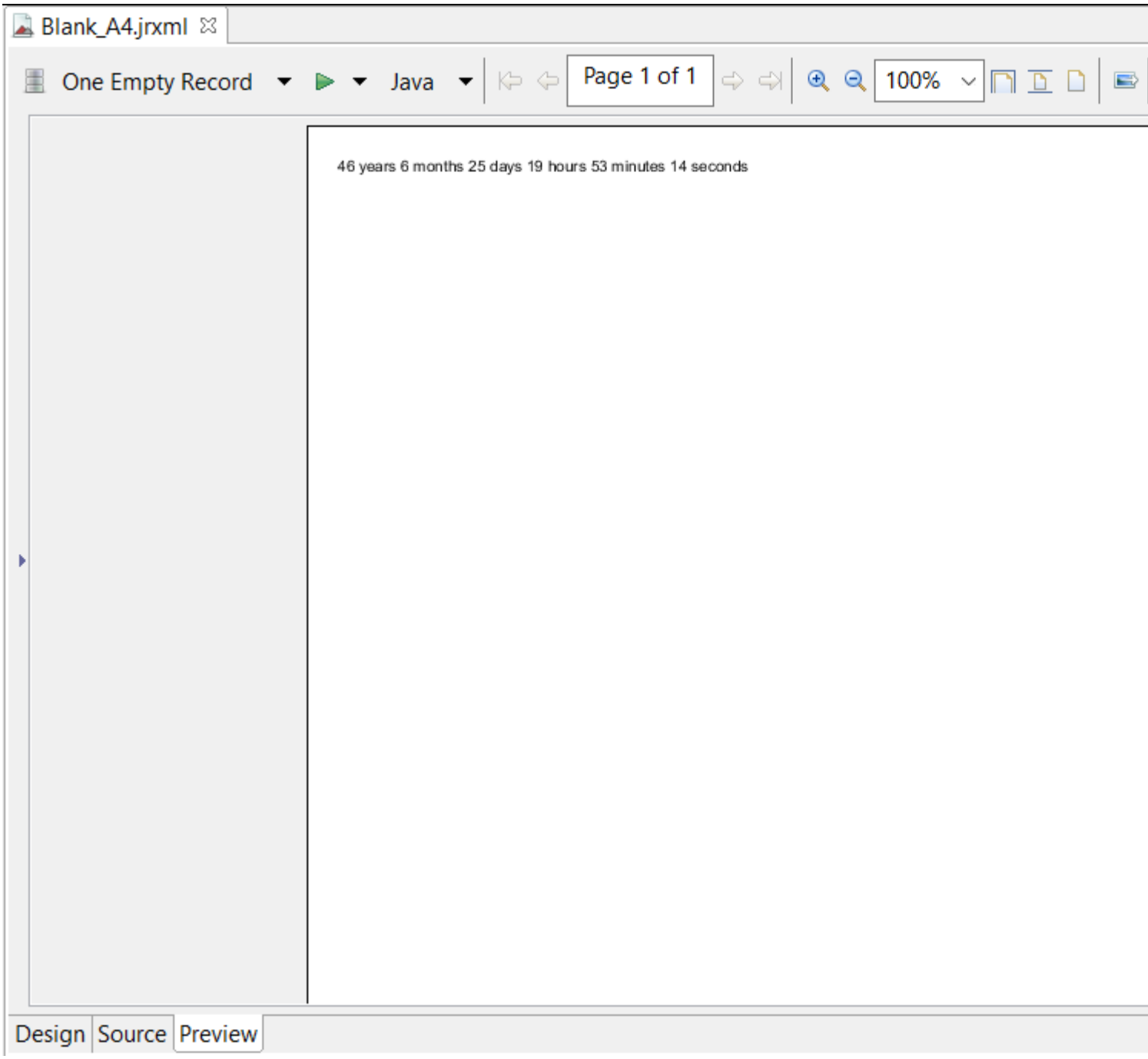
Para representar correctamente las **fuentes** en pdf, siempre se deben usar extensiones de fuentes (en classpath)

Examples

Con IDE (entorno de desarrollo integrado).

JasperSoft Studio

En Vista previa, ejecute el informe haciendo clic en la flecha verde, si no hay errores, se habilitará el menú de exportación, haga clic en el botón de exportación (imagen de disco) y seleccione "Exportar como PDF"



Con java

Para exportar una, debe completar el informe para obtener el objeto JasperPrint .

Exportar JasperPrint único (solo jrxml) al archivo

```
// 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();
```

Exportar múltiples JasperPrint's (multiple jrxml) a un solo archivo

Solo los primeros pasos difieren del conjunto anterior:

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

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

Los pasos restantes son los mismos:

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

Ver SimplePdfExporterConfiguration API para detalles de configuración.

Lea Exportar a pdf en línea: https://riptutorial.com/es/jasper-reports/topic/4190/exportar-a-pdf

Capítulo 4: Exportar a xls / xlsx

Examples

Con java

Exportar a 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();
}
```

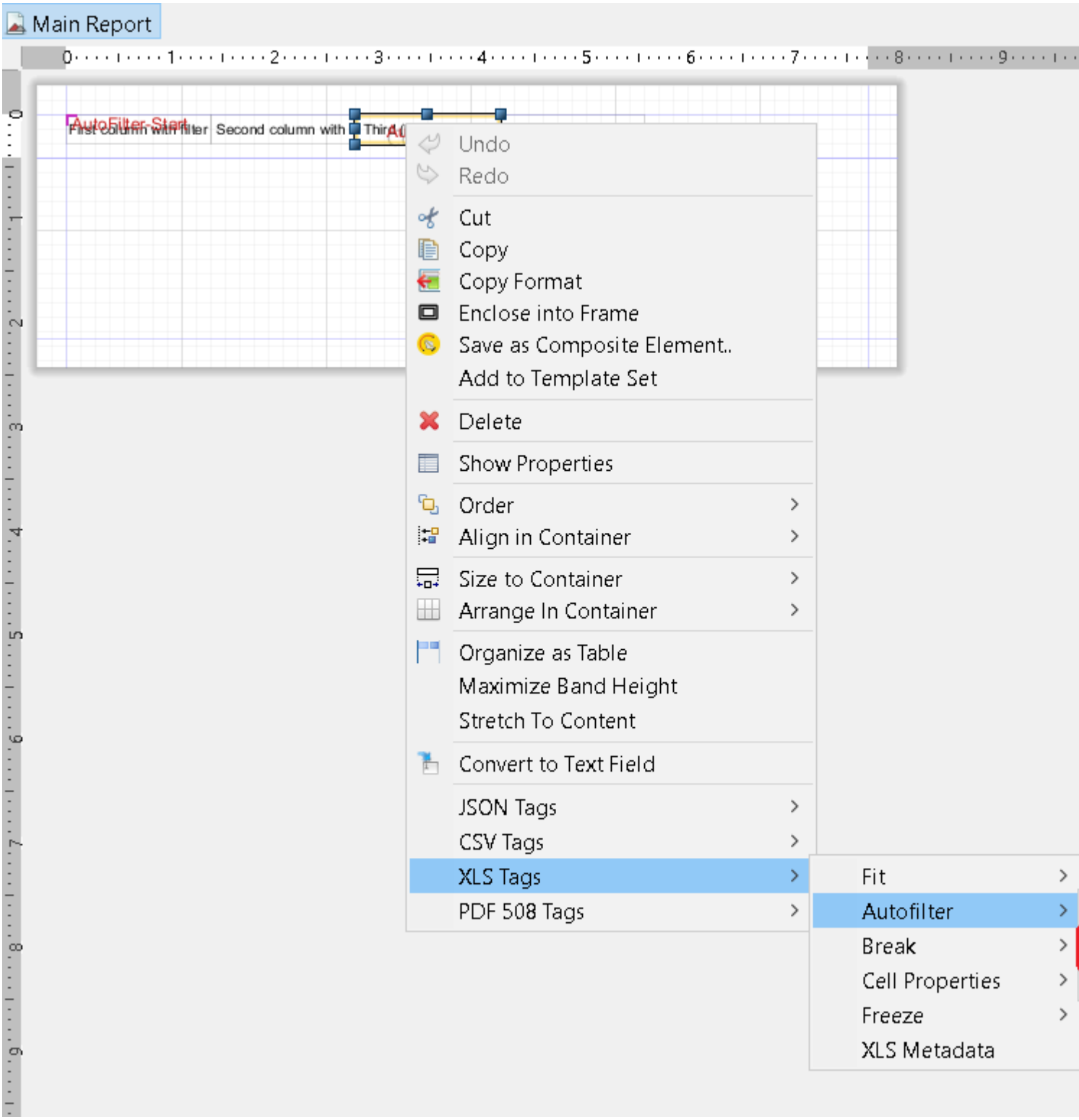
Añadiendo autofiltro para columnas

El uso de la propiedad ***net.sf.jasperreports.export.xls.auto.filter*** permite agregar autofiltro en el archivo xls generado.

```
<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 propiedad se puede configurar en *Jaspersoft Studio* con ayuda del menú contextual o manualmente editando el archivo *jrxml* .



Lea Exportar a xls / xlsx en línea: https://riptutorial.com/es/jasper-reports/topic/5008/exportar-a-

xls---xlsx

Capítulo 5: Extensiones de fuente

Examples

Creación y uso de extensiones de fuente

Crear una extensión de fuente utilizando el IDE. Consulte la documentación de iReport o Jaspersoft Studio para obtener más información. La extensión de la fuente también se puede crear manualmente .

¿Qué son las extensiones de fuente?

Usando un elemento de `textElement` puede especificar una fuente (si no se usa la fuente predeterminada, se usa `SansSerif`)

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

Para calcular la fuente métrica (para saltos de línea, alineación, etc.) y representar la fuente correctamente, la **fuente** debe **asignarse en la JVM** (Java virtual macchine). Podría instalar el archivo de fuente directamente en la JVM, pero esto no es alentador

De la última guía de JasperReport:

> Recomendamos encarecidamente a las personas que usen solo fuentes derivadas de extensiones de fuente, ya que esta es la única forma de asegurarse de que las fuentes estén disponibles para la aplicación cuando los informes se ejecuten en tiempo de ejecución. El uso de las fuentes del sistema siempre conlleva el riesgo de que los informes no funcionen correctamente cuando se implementan en una nueva máquina que podría no tener esas fuentes instaladas.

Extensión de fuente predeterminada

JasperReports proporciona una extensión de fuente predeterminada (ver maven distribution jasperreports-fonts.jar). Agregando esto a la ruta de clase, puede usar los siguientes nombres de fuente sin crear su propia extensión de fuente

> DejaVu Sans
> DejaVu Serif
> DejaVu Sans Mono

Problemas comunes

Cuestiones a considerar cuando se usan fuentes en pdf (itext):

- Al exportar a PDF, si el texto no se representa correctamente (faltan partes, no se muestran los caracteres, no se ajustan o no tienen el tamaño correcto), es probable que falten las **extensiones de fuente** .
- ¿ `.tff` **admite el** `.tff` real (OpenType) y la fuente puede **representar** realmente el carácter? No todas las fuentes representan todos los caracteres en `UTF-8` .
- ¿Se pasó la **codificación correcta** a iText? En caso de dudas (o en general) use la **codificación** `Identity-H` se recomienda para los nuevos estándares de PDF y le ofrece la posibilidad de mezclar diferentes codificaciones.
- ¿Está **incrustada** la fuente para que un PDF compartido en las computadoras pueda mostrar el contenido incluso si la fuente no está instalada? Si la fuente no es una de las 14 fuentes estándar de Tipo 1, insértela siempre.

Tenga en cuenta que la versión de iText utilizada por jasper report no procesará todas las fuentes (problema de ligaturizador). Puede probar la fuente `ttf` y la codificación directamente. ¿Cómo puedo probar si mi fuente está representada correctamente en pdf?

Lea Extensiones de fuente en línea: https://riptutorial.com/es/jasper-reports/topic/5773/extensiones-de-fuente

Capítulo 6: Llenar informe

Parámetros

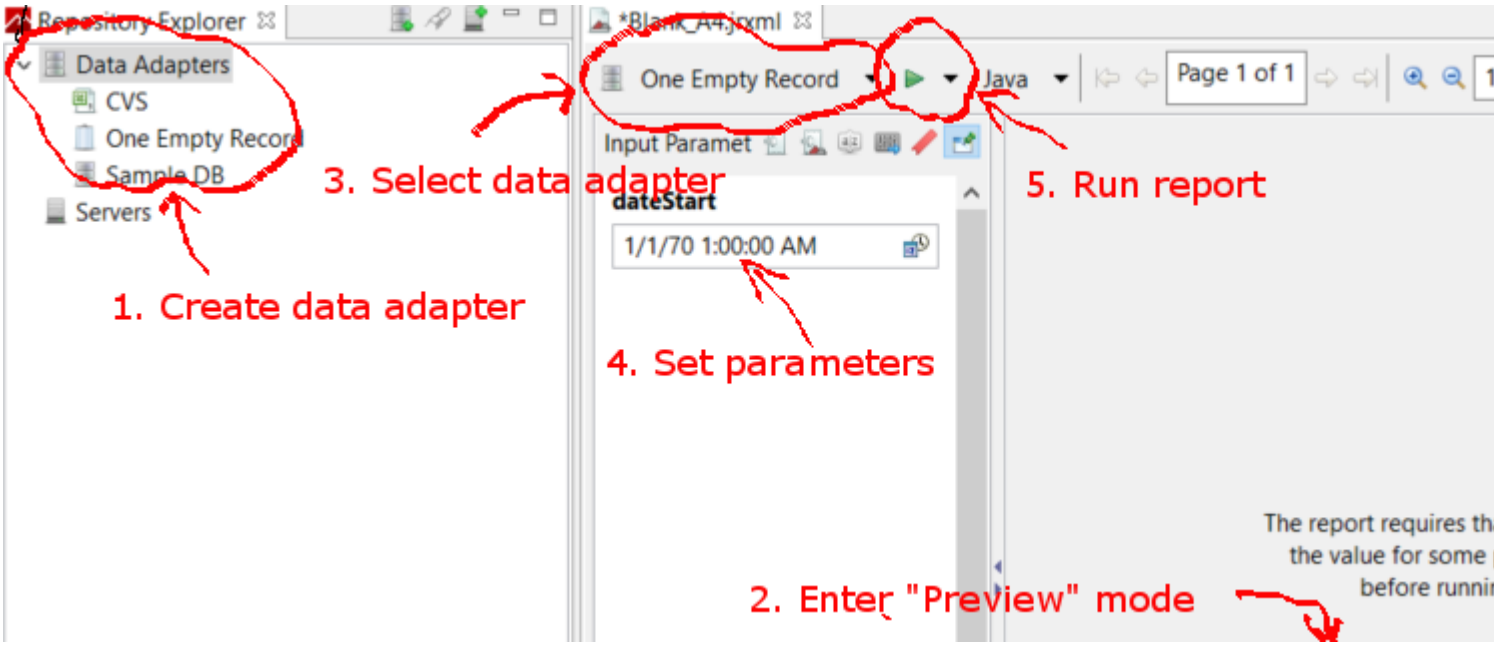
Parámetros	Columna
jasperimprimir	La salida del proceso de relleno que se puede exportar al formato deseado
Reportar plantilla	El archivo de diseño compilado `.jasper`
parámetros	El parámetro Mapa , que si está definido, puede ser referencia dentro del informe por `$P{key}`
fuente de datos	A net.sf.jasperreports.engine.JRDataSource
conexión	Una conexión de base de datos java.sql.Connection

Examples

Con IDE (entorno de desarrollo integrado).

JasperSoft Studio

1. Si se necesita una conexión de la fuente de datos o de la base de datos para completar el informe, cree su Adaptador de datos en Repository Explorer haciendo clic derecho en "Adaptadores de datos" seleccionando "Crear Adaptador de datos"
2. Ingrese al modo de vista previa seleccionando la pestaña **Vista previa** (no es necesario que haya errores en el diseño)
3. Seleccione la fuente de datos deseada (si no se requiere ninguna fuente de datos, seleccione "Un registro vacío")
4. Configure el parámetro como desee
5. Rellene el informe haciendo clic en la flecha verde "Ejecutar el informe"



Rellene la plantilla de JasperReport usando Java

Requisitos comunes

Todos los informes, independientemente de cómo se presenten los datos, toman una ruta hacia la plantilla de informe y un mapa de parámetros. Las variables se utilizan en todos los ejemplos que siguen:

```
// 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";
```

El uso de un archivo `.jrxml` un paso de compilación adicional que no es necesario en la mayoría de las situaciones. A menos que haya escrito un software personalizado para cambiar el `.jrxml` antes de que se `.jrxml` el informe (por ejemplo, agregar o eliminar columnas dinámicamente), use el archivo `.jasper` como se muestra en los ejemplos siguientes.

Usando una conexión de base de datos

```
// 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);
```

Usando una fuente de datos personalizada

```
// 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);
```

Sin fuente de datos, banda de detalle no utilizada

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

> Sin una `whenNoDataType="AllSectionsNoDetail"` datos, el atributo `whenNoDataType="AllSectionsNoDetail"` en el elemento `JasperReport` se debe establecer, de lo contrario se generará un informe vacío (en blanco).

Lea Llenar informe en línea: https://riptutorial.com/es/jasper-reports/topic/3958/llenar-informe

Capítulo 7: Usando subinformes

Parámetros

Parámetro	Detalles
parametersMapExpression	El mapa con parámetros. *No requerido*
subinformeParámetro	El par de nombre y valor (establecido con ***subreportParameterExpression***). *No es necesario* . Varios parámetros se pueden pasar al subinforme
expresión de conexión	Conexión para obtener datos. *No requerido*
dataSourceExpression	Expresión para pasar la fuente de datos. *No requerido*
subreportExpression	La ruta del subinforme / URI o incluso el objeto JasperReport. *No requerido*
retornoValor	El par de nombre y valor. *No es necesario* . Se pueden devolver varios valores del subinforme al informe maestro.

Observaciones

- Los subinformes se pueden usar para construir informes complejos. La reutilización de los informes existentes es otro objetivo del uso de subinformes.
- El subinforme se mostrará como parte del informe maestro en caso de usar el elemento `<subreport>` .
- El valor del parámetro ***subreportExpression*** es diferente para usar en *JasperReports Server* o simplemente por el marco de *JasperReports* (algunas *API se* usan o usan en IDE).

  Para el *servidor JasperReports* parece:

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

  Para usar solo con el motor *JasperReports* :

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

  La gran explicación de *@AndreasDietrich* se puede encontrar en *JasperServer: No se puede encontrar la* publicación de *excepción de subinforme*

- Por algunas razones, el subinforme se puede usar como un informe común, sin llamar desde el informe maestro (con la ayuda del elemento `<subreport>`). El subinforme es

siempre un informe.

Examples

Pasando la conexión al subinforme; devuelve los valores al informe maestro

Este es un fragmento del informe maestro. Dos parámetros y la conexión (por ejemplo, *jdbc*) pasan al subinforme. Se devuelve un valor desde el subinforme al informe maestro, este valor (*variable*) se puede usar en el informe maestro

```
<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>
```

Pasar datasoure a subinforme

Este es un fragmento del informe maestro. El origen de datos se pasa al subinforme con la ayuda del constructor ***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>
```

Lea Usando subinformes en línea: https://riptutorial.com/es/jasper-reports/topic/5452/usando-subinformes

Creditos

S. No	Capítulos	Contributors
1	Comenzando con los informes de jaspe	Alex K, Community, Dave Jarvis, Petter Friberg
2	Compile JasperReports .jrxml a .jasper	Alex K, Dave Jarvis, Petter Friberg
3	Exportar a pdf	Alex K, Dave Jarvis, Petter Friberg, RamenChef
4	Exportar a xls / xlsx	Alex K
5	Extensiones de fuente	Dave Jarvis, Petter Friberg
6	Llenar informe	Alex K, Dave Jarvis, Petter Friberg
7	Usando subinformes	Alex K