



Creación de informes con JasperReports



Diseño de informes en JasperReports

Para la creación de informes en Java podemos usar JasperReports utilizando una aplicación llamada [JasperSoft Studio](#).

Una vez descargado e instalado tendremos un programa para la creación de informes similar a cualquier IDE que usemos para crear programas en Java.



Reposi... × Project...

- ▼ Data Adapters
 - New Data Adapter
 - One Empty Record
 - Sample DB
- Servers

Outline ×

There is no active editor that provides an outline.

Report State ×

Console Errors Statistics

Palette ×

A palette is not available.

Prop... ×

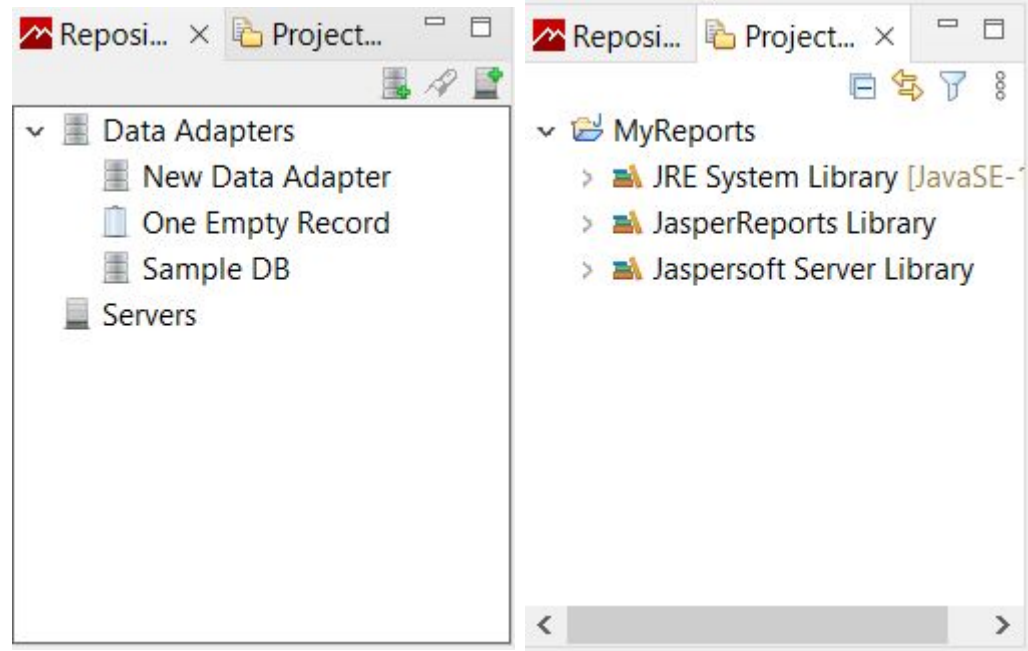
Prob...

Property

Value

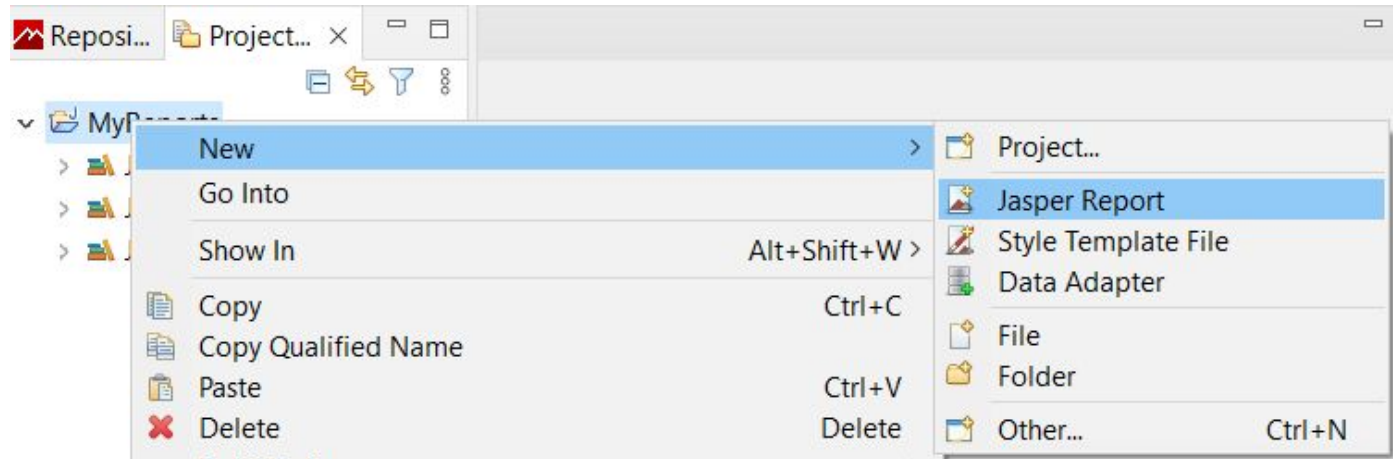
Diseño de informes en JasperReports

En la parte izquierda veremos dos secciones, una para gestionar las conexiones de datos (Data Adapters) que se pueden usar y otro para la gestión de proyectos, donde crearemos los informes.

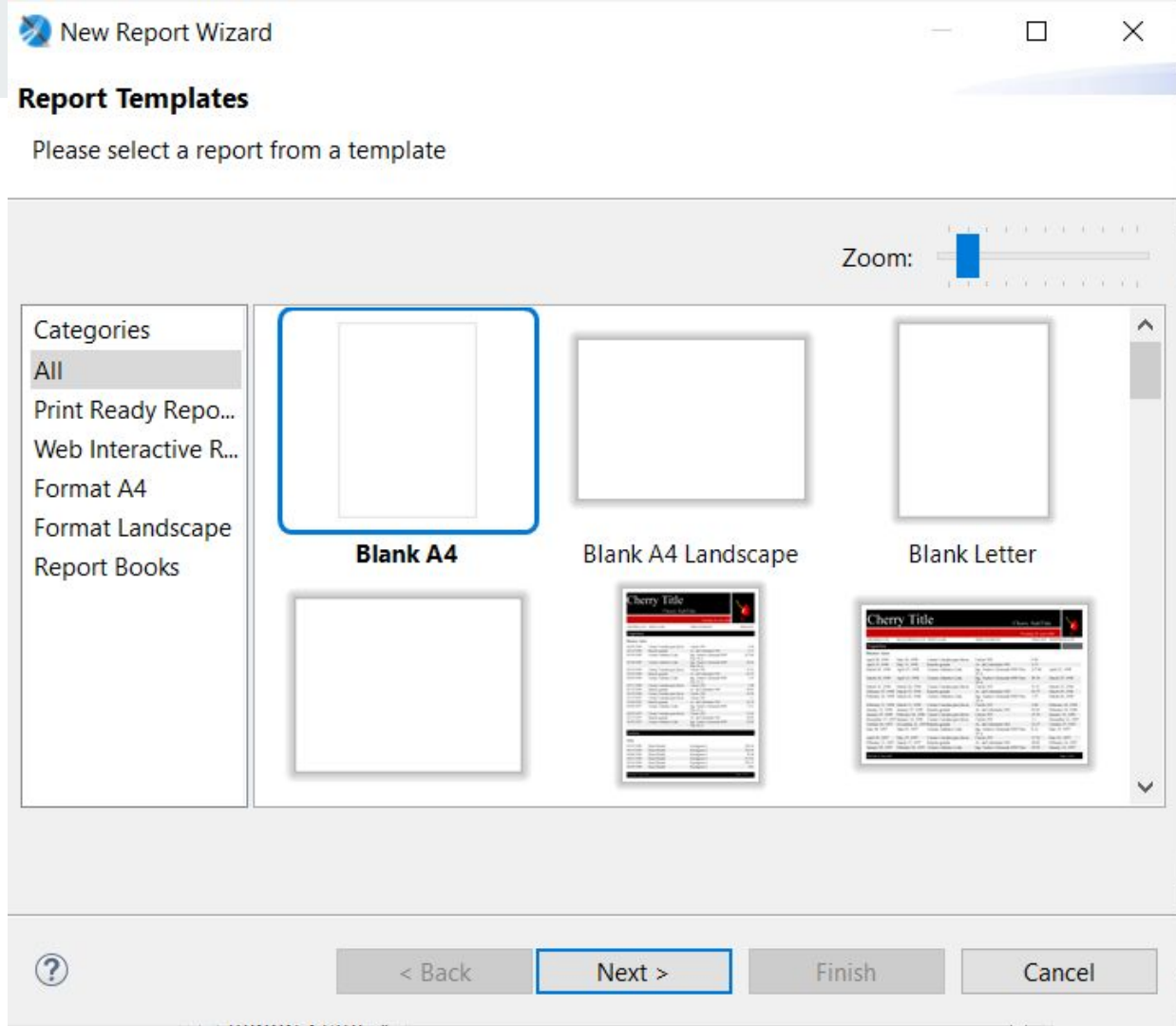


Diseño de informes en JasperReports

Para crear un informe pulsaremos con el segundo botón sobre el proyecto -> Nuevo -> Jasper Report



Se nos mostrará una pantalla donde podremos elegir una plantilla para la creación del informe.



Indicaremos un nombre para el informe con extensión **jrxml**

New Report Wizard

Report file

Please select your reports file name with .jrxml extension.

Enter or select the parent folder:

MyReports

> MyReports

File name: PrimerInforme.jrxml

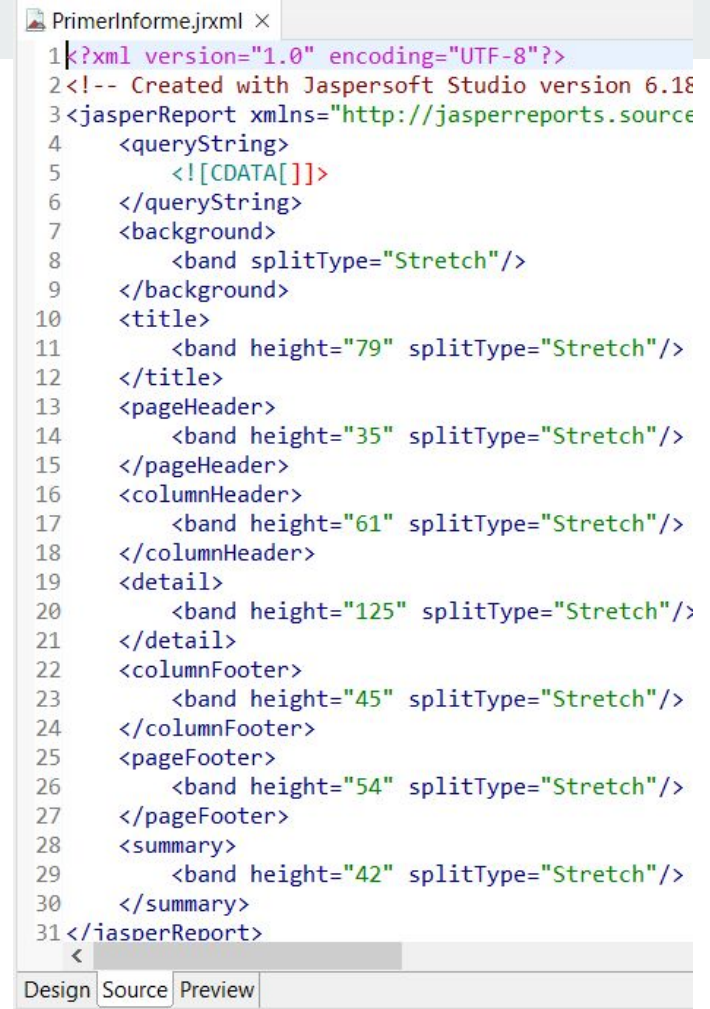
? < Back Next > Finish Cancel

The screenshot shows the 'Main Report' in design view. The layout is structured as follows:

- Title**: Located at the top center of the report area.
- Page Header**: Located below the title, spanning the width of the report.
- Column Header**: Located below the page header, spanning the width of the report.
- Detail 1**: A large section below the column header, consisting of multiple rows.
- Column Footer**: Located below the detail section, spanning the width of the report.
- Page Footer**: Located below the column footer, spanning the width of the report.
- Summary**: The final section at the bottom of the report.

The bottom navigation bar shows the 'Design' tab is active, with 'Source' and 'Preview' tabs also visible.

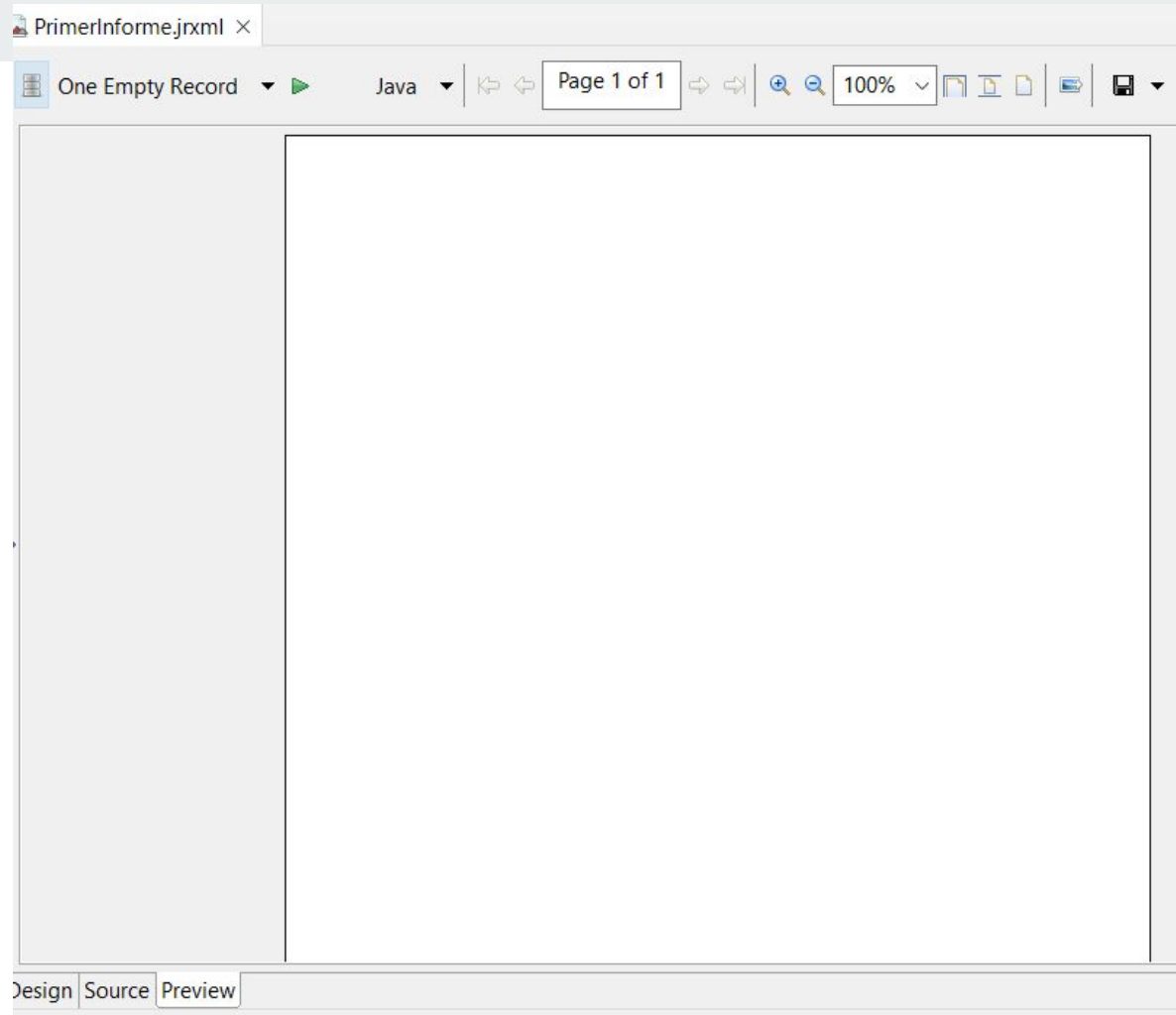
También podremos ver y modificar el código del informe en xml



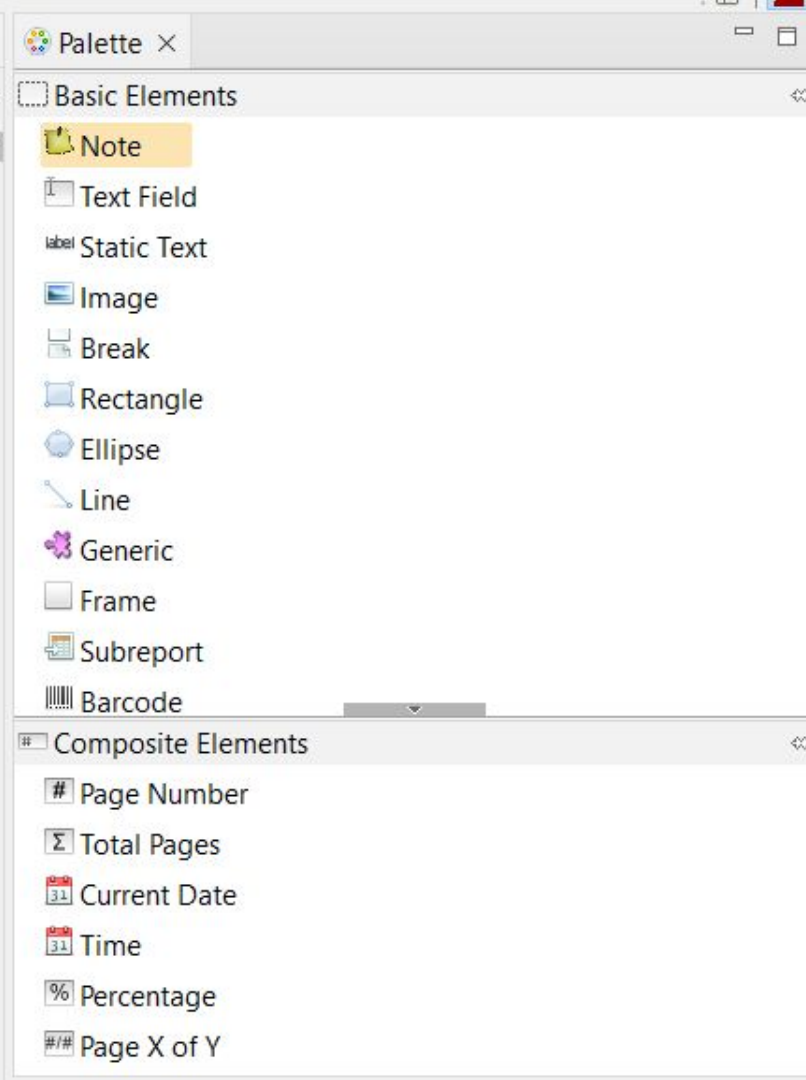
```
1<?xml version="1.0" encoding="UTF-8"?>
2<!-- Created with JasperSoft Studio version 6.18
3<jasperReport xmlns="http://jasperreports.sourceforge
4  <queryString>
5    <![CDATA[]]>
6  </queryString>
7  <background>
8    <band splitType="Stretch"/>
9  </background>
10 <title>
11   <band height="79" splitType="Stretch"/>
12 </title>
13 <pageHeader>
14   <band height="35" splitType="Stretch"/>
15 </pageHeader>
16 <columnHeader>
17   <band height="61" splitType="Stretch"/>
18 </columnHeader>
19 <detail>
20   <band height="125" splitType="Stretch"/>
21 </detail>
22 <columnFooter>
23   <band height="45" splitType="Stretch"/>
24 </columnFooter>
25 <pageFooter>
26   <band height="54" splitType="Stretch"/>
27 </pageFooter>
28 <summary>
29   <band height="42" splitType="Stretch"/>
30 </summary>
31</jasperReport>
```

Design Source Preview

Y tenemos una pestaña de previsualización del informe donde podemos ver cómo va quedando.



En la parte de la derecha tenemos la típica paleta de componentes donde se muestra aquellos elementos que podemos tirar en el informe.






Diseño de informes en JasperReports

Para la ejecución del informe en un proyecto de Java necesitaremos una serie de librerías externas que podemos obtener en el repositorio de Maven (mucho más sencillo obtenerlos en un proyecto Maven, al descargarse automáticamente), en la página de JasperReports mediante registro, o en el repositorio de jar:

<https://jar-download.com/artifact-search/jasperreports>



Una vez descargados podemos añadirlos al proyecto en una carpeta lib (y referenciarlos dentro del proyecto)

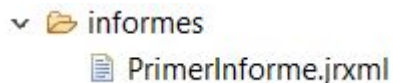
▼ lib

- bcprov-jdk15on-1.68.jar
- castor-core-1.4.1.jar
- castor-xml-1.4.1.jar
- commons-beanutils-1.9.4.jar
- commons-collections-3.2.2.jar
- commons-collections4-4.2.jar
- commons-digester-2.1.jar
- commons-lang3-3.4.jar
- commons-logging-1.1.1.jar
- ecj-3.21.0.jar
- itext-2.1.7.js9.jar
- jackson-annotations-2.12.2.jar
- jackson-core-2.12.2.jar
- jackson-databind-2.12.2.jar
- jasperreports-6.18.1.jar
- javax.inject-1.jar
- jcommon-1.0.23.jar
- jfreechart-1.0.19.jar



Diseño de informes en JasperReports

Una vez añadidas las librerías necesarias lo siguiente que podemos hacer es incluir el informe en el proyecto, por ejemplo en una carpeta de informes (aunque también podemos tener un directorio compartido en local o en red donde varias aplicaciones accedan al informe).





Diseño de informes en JasperReports

Lo último será ejecutar el informe. Para ello se realizan tres acciones: Compilar el informe, rellenarlo de datos y visualizarlo.

```
private void mostrarInforme() {
    String informe="./informes/PrimerInforme.jrxml";

    try {
        //Compilar el informe
        JasperReport report = JasperCompileManager.compileReport(informe);

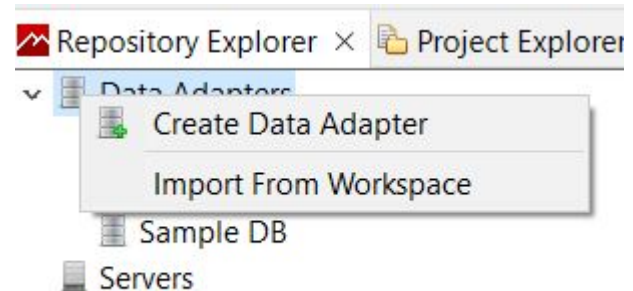
        //Rellenar el informe
        JasperPrint visor= JasperFillManager.fillReport(report, null, new JREmptyDataSource());

        //Visualizar el informe
        JasperViewer.viewReport(visor, false);

    } catch (JRException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

Conexión con base de datos en JasperReports

Para realizar una conexión con datos de una BBDD lo primero que podemos hacer es crear un DataAdapter en la pestaña correspondiente:



Se muestra una pestaña donde deberemos indicar:

- El Driver que se va a usar.
- La cadena de conexión.
- El usuario para la conexión.
- El password de conexión.

Data Adapter Wizard

Data Adapter

Database JDBC Connection

Name:

JDBC Driver:

JDBC Url:

Username:

Password:

Attention! Passwords are saved in clear text

Database Location | Connection Properties | Driver Classpath

? Test < Back Next > Finish Cancel

En la pestaña de Driver, para realizar pruebas, deberemos indicar la localización del Driver a usar.

Data Adapter Wizard

Data Adapter

Database JDBC Connection

Name:

Jar Files Path

C:\Users\Fernando\Downloads\mysql-connector-java-8.0.29.jar

Add

Delete

Database Location | Connection Properties | **Driver Classpath**

? Test < Back Next > **Finish** Cancel

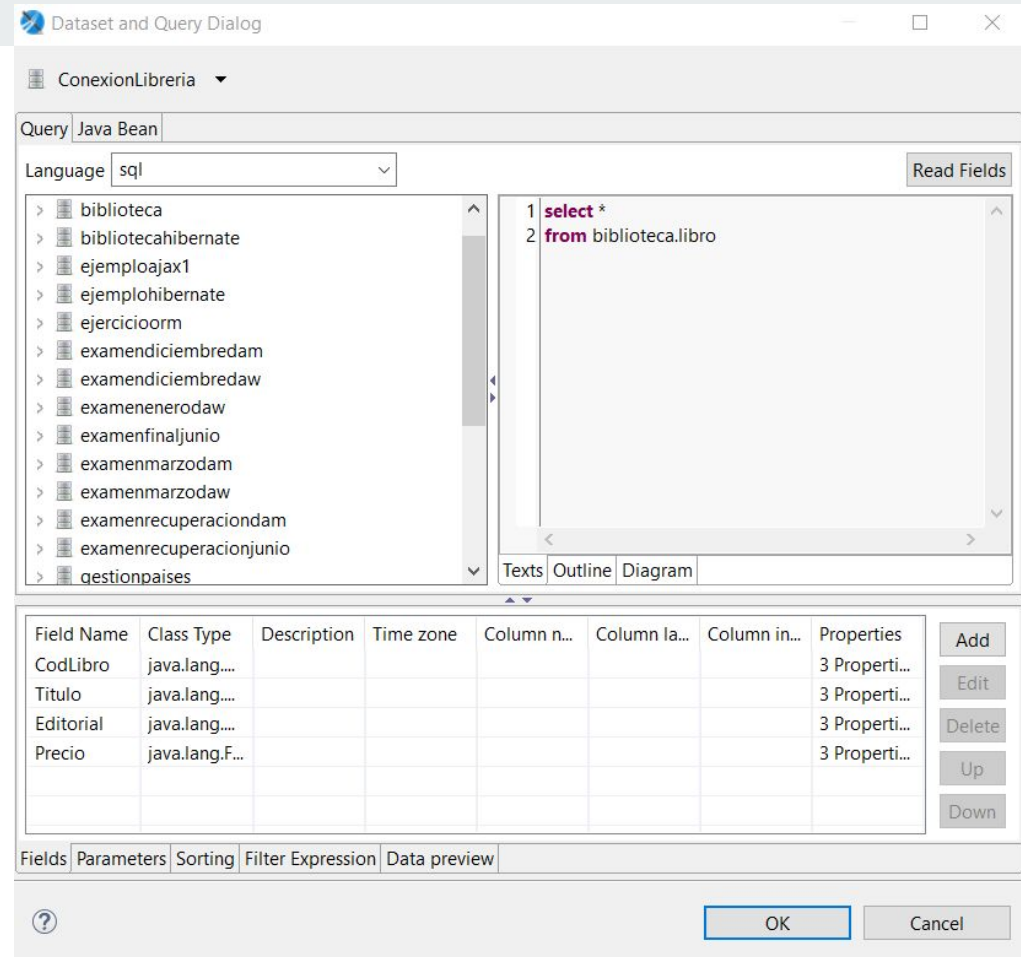
Conexión con base de datos en JasperReports

En la parte superior del informe en edición, podemos ver un botón para editar el DataSet e indicar la consulta del informe.



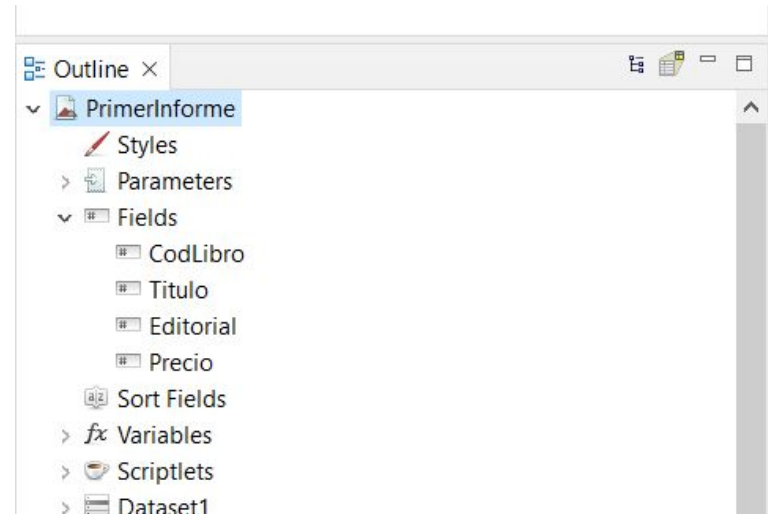
Al pulsar el botón se nos muestra un editor donde deberemos escoger el DataAdapter creado y a partir de ahí podremos crear una consulta sobre cualquier base de datos accesible mediante dicha conexión.

En la parte inferior, tras realizar la consulta, podremos ver los campos consultados.



Conexión con base de datos en JasperReports

Después de crear el DataSet podremos ver en la parte de la derecha los campos vinculados.





Conexión con base de datos en JasperReports

Estos campos los podemos tirar en el informe, por ejemplo en el apartado de detalles, para que se muestren para cada registro de la consulta, indicando en la cabecera de detalles el título que aparecerá.

Ejemplo de informe		
Title		
Page Header		
Column Header		
Titulo	Editorial	Precio
\$F{Titulo}	\$F{Editorial}	\$F{Precio}
Detail 1		
Column Footer		
Page Footer		
Summary		



Conexión con base de datos en JasperReports

Al previsualizar el informe veremos los datos con la estructura indicada.

Ejemplo de informe

Título	Editorial	Precio
Los asesinos del emperador	Planeta	20.0
La caída de los gigantes	De Bolsillo	15.0
Violetas de Marzo	RBA	9.0



Conexión con base de datos en JasperReports

Para poder visualizar el informe desde el proyecto Java necesitaremos una conexión (y el driver de conexión) en el proyecto.

Después de tenerlos indicaremos la conexión en la parte de rellenar los datos del informe.



En el relleno indicamos
la conexión activa en el
método *fillReport*

```
private Connection obtenerConexion() throws ClassNotFoundException, SQLException {
    Class.forName("org.mariadb.jdbc.Driver");
    String url = "jdbc:mariadb://localhost:3306/biblioteca";
    Connection con = DriverManager.getConnection(url, "root", "");
    return con;
}

private void mostrarInforme() {
    String informe = "./informes/PrimerInforme.jrxml";

    try {
        //Compilar el informe
        JasperReport report = JasperCompileManager.compileReport(informe);

        //Rellenar el informe
        JasperPrint visor = JasperFillManager.fillReport(report, null, obtenerConexion());

        //Visualizar el informe
        JasperViewer.viewReport(visor, false);
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

Project Explorer

JasperViewer

Ver informe

Ejemplo de informe

Título	Editorial	Precio
Los asesinos del emperador	Planeta	20.0
La caída de los gigantes	De Bolsillo	15.0
Violetas de Marzo	RBA	9.0

entidades

- GestionLibros.java
- vista.pantallas
 - PantallaDatos.java

Referenced Libraries

- mariadb-java-client-3.0.3.jar - C:\l

EjemploDespliegueProperties

EjemploInformes

JRE System Library [JavaSE-1.8]

src

Referenced Libraries

- informes
- lib

EjemploJavaHelp



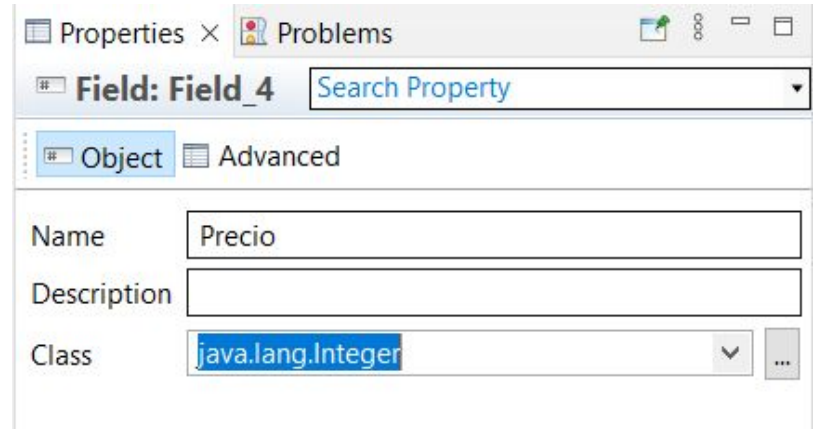
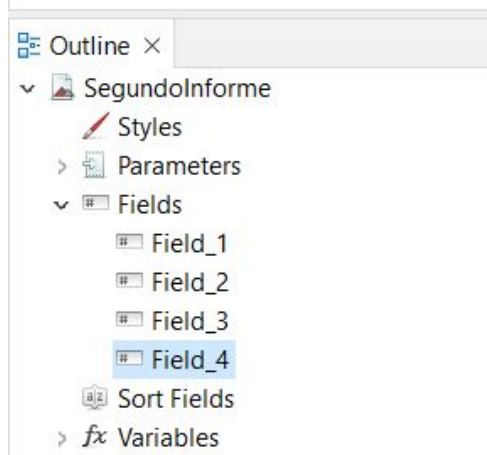
Conexión con base de datos en JasperReports 2

En lo visto anteriormente se define la consulta mediante el DataSet dentro del informe y después se le pasa la conexión activa.

Es posible realizarlo de otra forma, indicando en el proyecto Java tanto la conexión como la consulta y que el informe tan solo sirva para representar los datos.

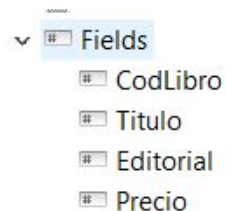
Conexión con base de datos en JasperReports 2

Lo primero que tendremos que hacer es crear los campos y renombrarlos indicando también el tipo de datos.



Conexión con base de datos en JasperReports 2

Una vez renombrados los usamos en el informe de la misma manera a la vista anteriormente.



Title			
Page Header			
CodLibro	Titulo	Editorial	Precio
Column Header			
\$F{CodLibro}	\$F{Titulo}	\$F{Editorial}	\$F{Precio}
Detail 1			
Column Footer			
Page Footer			
Summary			



Conexión con base de datos en JasperReports 2

En el código indicaremos la consulta (los nombres deben coincidir con los campos del informe).

Modificaremos el método fill en el que le pasaremos, en vez de una conexión, un objeto de tipo JRResultSetDataSource con los datos de la consulta.

```
String informe="./informes/SegundoInforme.jrxml";

try {
    Connection conexion= obtenerConexion();
    Statement st= conexion.createStatement();
    ResultSet rs= st.executeQuery("SELECT CodLibro, Titulo, Editorial,Precio FROM `libro`");

    JRResultSetDataSource ds= new JRResultSetDataSource(rs);

    //Compilar el informe
    JasperReport report = JasperCompileManager.compileReport(informe);

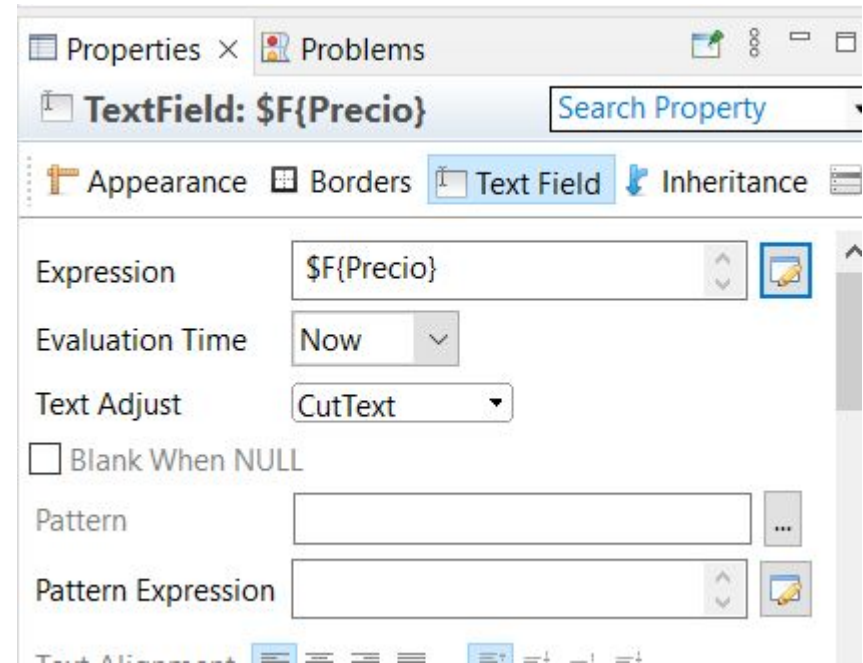
    //Rellenar el informe
    JasperPrint visor= JasperFillManager.fillReport(report, null, ds);

    //Visualizar el informe
    JasperViewer.viewReport(visor,false);

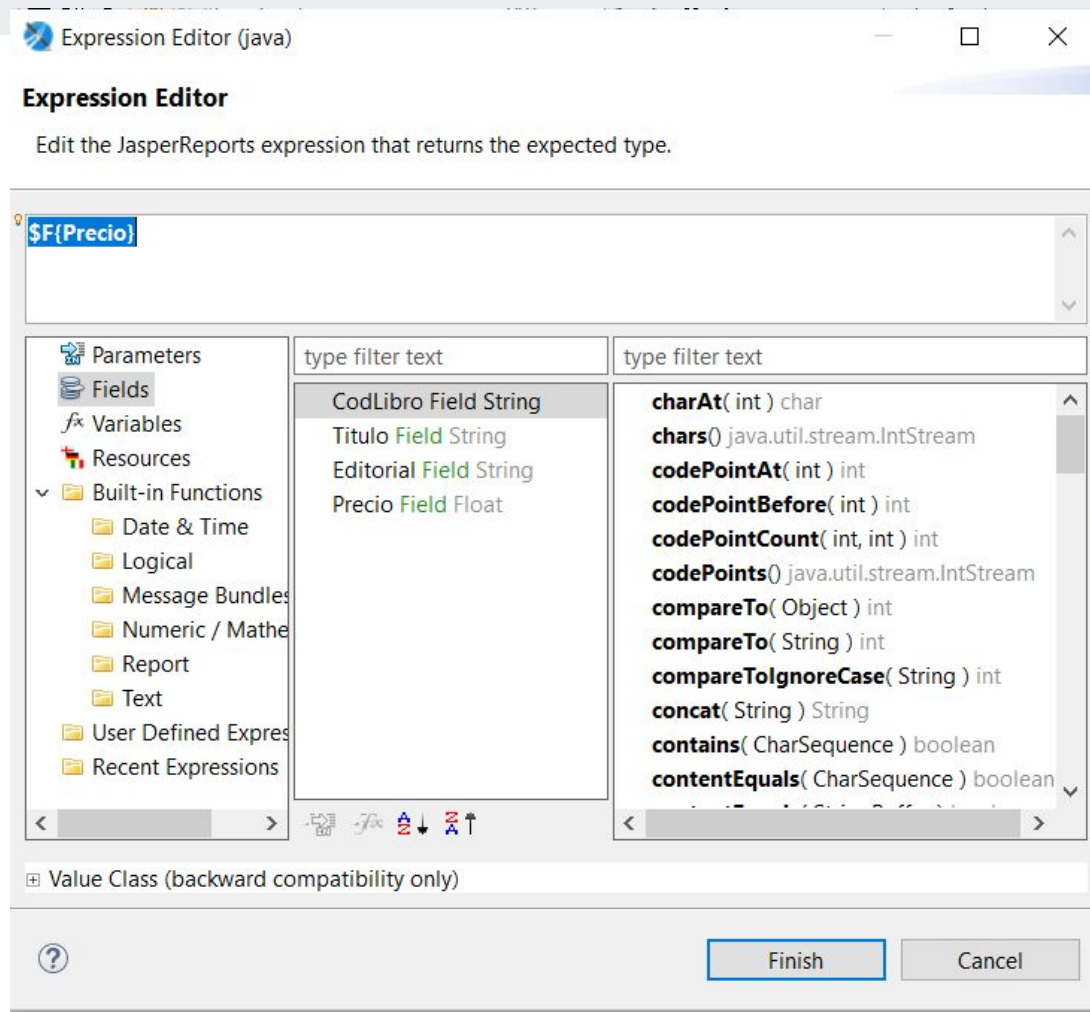
} catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

Expresiones de campos en JasperReports

Entre las propiedades de un campo de texto se puede indicar una expresión para la visualización de los datos.



Se pueden utilizar funciones a la hora de mostrar los datos, por ejemplo en precio, si queremos que los valores menores de 10 salgan con valor 0 podremos usar un if en su forma simplificada:



Expression Editor

Edit the JasperReports expression that returns the expected type.

(\$F{Precio} < 10)?0:\$F{Precio}

	type filter text	type filter text
Parameters		
Fields	CodLibro Field String	charAt (int) char
Variables	Titulo Field String	chars () java.util.stream.IntStream
Resources	Editorial Field String	codePointAt (int) int
Built-in Functions	Precio Field Float	codePointBefore (int) int
Date & Time		codePointCount (int, int) int
Logical		codePoints () java.util.stream.IntStream
Message Bundles		compareTo (Object) int
Numeric / Mathe		compareTo (String) int
Report		compareToIgnoreCase (String) int
Text		concat (String) String
User Defined Express		contains (CharSequence) boolean
Recent Expressions		contentEquals (CharSequence) boolean

Value Class (backward compatibility only)

Si tenemos este if:

```
if(precio<10){
    valor=0;
else
    valor= precio;
```


Su versión simplificada es:

```
valor= (precio<10)?0:precio;
```



Finish

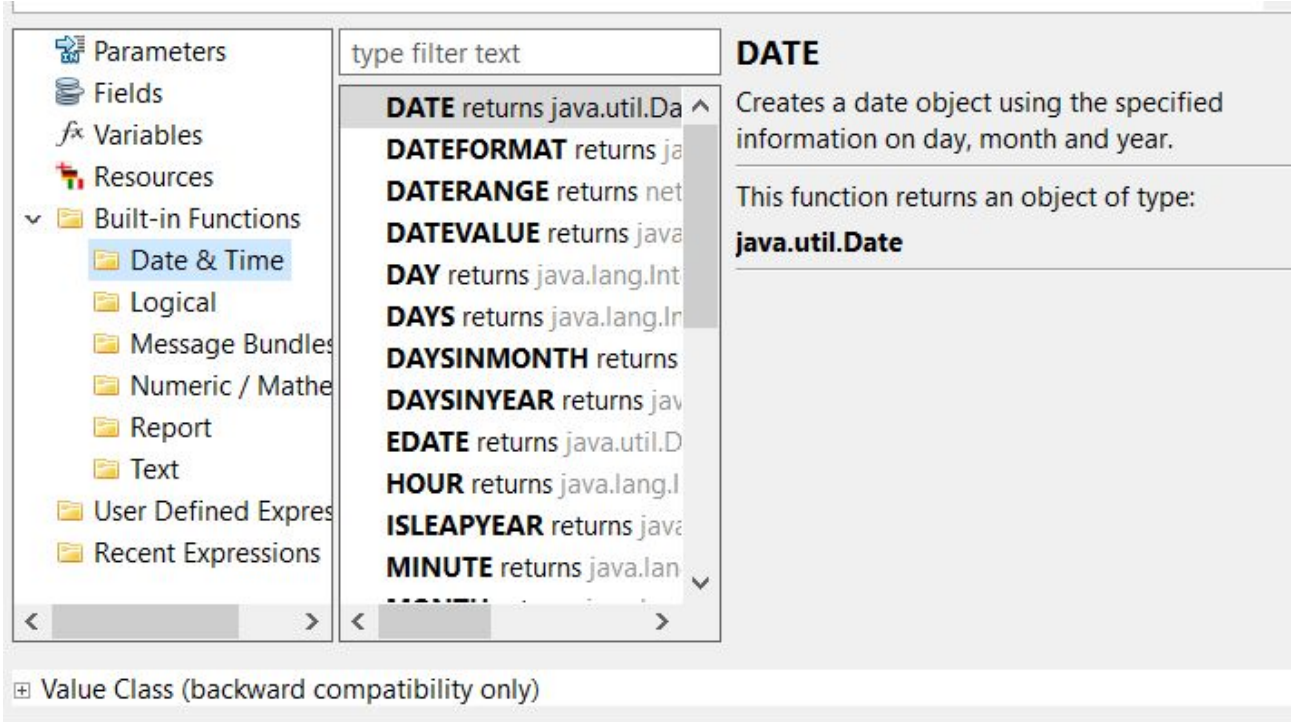
Cancel



Ejemplo de informe

Título	Editorial	Precio
Los asesinos del emperador	Planeta	20.0
La caída de los gigantes	De Bolsillo	15.0
Violetas de Marzo	RBA	0.0

Además tenemos a disposición múltiples funciones para el tratamiento de los datos.



Parameters

Fields

Variables

Resources

Built-in Functions

- Date & Time
- Logical
- Message Bundles
- Numeric / Mathe
- Report
- Text
- User Defined Expres
- Recent Expressions

type filter text

DATE returns java.util.Date

DATEFORMAT returns ja

DATERANGE returns net

DATEVALUE returns java

DAY returns java.lang.Int

DAYS returns java.lang.In

DAYSINMONTH returns

DAYSINYEAR returns jav

EDATE returns java.util.D

HOUR returns java.lang.l

ISLEAPYEAR returns java

MINUTE returns java.lan

DATE

Creates a date object using the specified information on day, month and year.

This function returns an object of type:

java.util.Date

+ Value Class (backward compatibility only)

Formato de campos en JasperReports

También podemos indicar el formato de representación de los campos mediante el campo ***Pattern*** en las propiedades del campo.

The screenshot shows the 'Text Field Properties' dialog box in JasperReports. The title bar reads 'TextFie...Precio}' and there is a 'Search Property' search bar. Below the title bar are four tabs: 'Appearance' (selected), 'Borders', 'Text Field', and 'Inheritance'. The 'Text Adjust' property is set to 'CutText'. There is a checkbox for 'Blank When NULL' which is currently unchecked. The 'Pattern' property has an empty text field and a button with three dots. The 'Pattern Expression' property has an empty text field and a button with a document icon. At the bottom, there are sections for 'Text Alignment' with icons for left, right, center, and justified alignment, and 'Rotation' with icons for 'St-', 0°, 90°, and 180°.

TextFie...Precio} Search Property

Appearance Borders Text Field Inheritance

Text Adjust CutText

☐ Blank When NULL

Pattern

Pattern Expression

Text Alignment

Rotation St- 0° 90° 180°

Se puede indicar el tipo de datos con el formato a usar.



Pattern

Format Pattern

Specify format pattern, in java style.

Number
Percentage
Currency
Scientific
Time
Date

Pattern

Leading zeros

Decimal places:

☐ Use 1000 separator

€-10.023,12
10.023,12- €
(10.023,12) €
€(-10.023,123)
€(10.023,123-)

Currency format is used to display monetary values.



Finish

Cancel



Paso de parámetros en JasperReport

El filtrado de datos normalmente se realizará del lado de la aplicación Java, pero podemos pasarle valores al informe en forma de parámetros para que quede reflejado con qué valores se sacó dicho informe.

Para poder realizarlo, lo primero será definir dentro del informe el parámetro en su correspondiente sección.

Parameters

- REPORT_CONTEXT
- REPORT_PARAMETERS_MAP
- JASPER_REPORTS_CONTEXT
- JASPER_REPORT
- REPORT_CONNECTION
- REPORT_MAX_COUNT
- REPORT_DATA_SOURCE
- REPORT_SCRIPTLET
- REPORT_LOCALE
- REPORT_RESOURCE_BUNDLE
- REPORT_TIME_ZONE
- REPORT_FORMAT_FACTORY
- REPORT_CLASS_LOADER
- REPORT_TEMPLATES
- SORT_FIELDS
- FILTER
- REPORT_VIRTUALIZER
- IS_IGNORE_PAGINATION
- Parameter1

Properties × Problems

Parame..._LIBRO

☒ Object ☐ Advanced

Name

Class

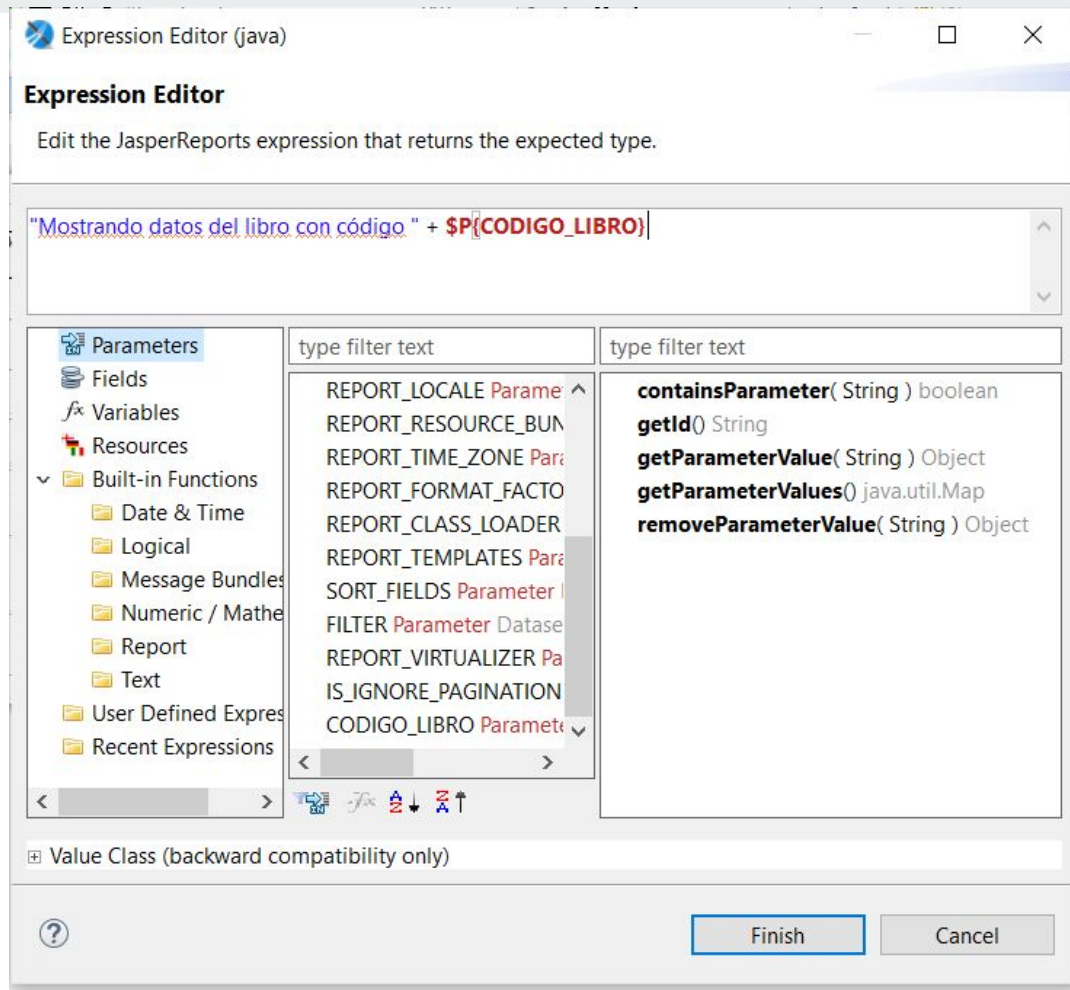
Description

☒ Is For Prompting

Default Value Expression

Evaluation Time

Después podemos crear un campo de texto indicando una expresión que contenga al parámetro, por ejemplo la que se muestra en el pantallazo:





Paso de parámetros en JasperReport

Tras la definición y uso del parámetro dentro del informe deberemos usarlo en la aplicación Java.

El siguiente método muestra su uso desde un método que recibe un valor para el código de un libro

```
//Filtrado de la consulta con el valor del parámetro
PreparedStatement st= conexion.prepareStatement("
    + "SELECT CodLibro, Titulo, Editorial,Precio "
    + "FROM libro "
    + "where libro.CodLibro = ? ");
st.setString(1, codigo);
ResultSet rs= st.executeQuery();

JRResultSetDataSource ds= new JRResultSetDataSource(rs);
```



Paso de parámetros en JasperReport

Después definimos un objeto de tipo HashMap para almacenar los parámetros del informe a invocar.

```
//Se define un HashMap para almacenar los parámetros ("nombre en el informe" - "valor")  
HashMap<String,Object> parametros= new HashMap<String,Object>();  
parametros.put("CODIGO_LIBRO", codigo);
```



Paso de parámetros en JasperReport

Después haríamos las tres acciones sobre informes: Compilar, Rellenar y Visualizar, modificando ahora la parte de rellenar para indicarle los valores de parámetros.

```
//Compilar el informe
JasperReport report = JasperCompileManager.compileReport(informe);

//Rellenar el informe
//En el fillReport le pasamos el HashMap de los parámetros
JasperPrint visor= JasperFillManager.fillReport(report, parametros, ds);

//Visualizar el informe
JasperViewer.viewReport(visor, false);
```



JasperViewer

Mostrando datos del libro con código LAE01

CodLibro	Titulo	Editorial	Precio
LAE01	Los asesinos del emperador	Planeta	20



Insertión de imágenes en JasperReport



Subinformes en JasperReport



Uso de variables en JasperReports



Diagramas en JasperReports