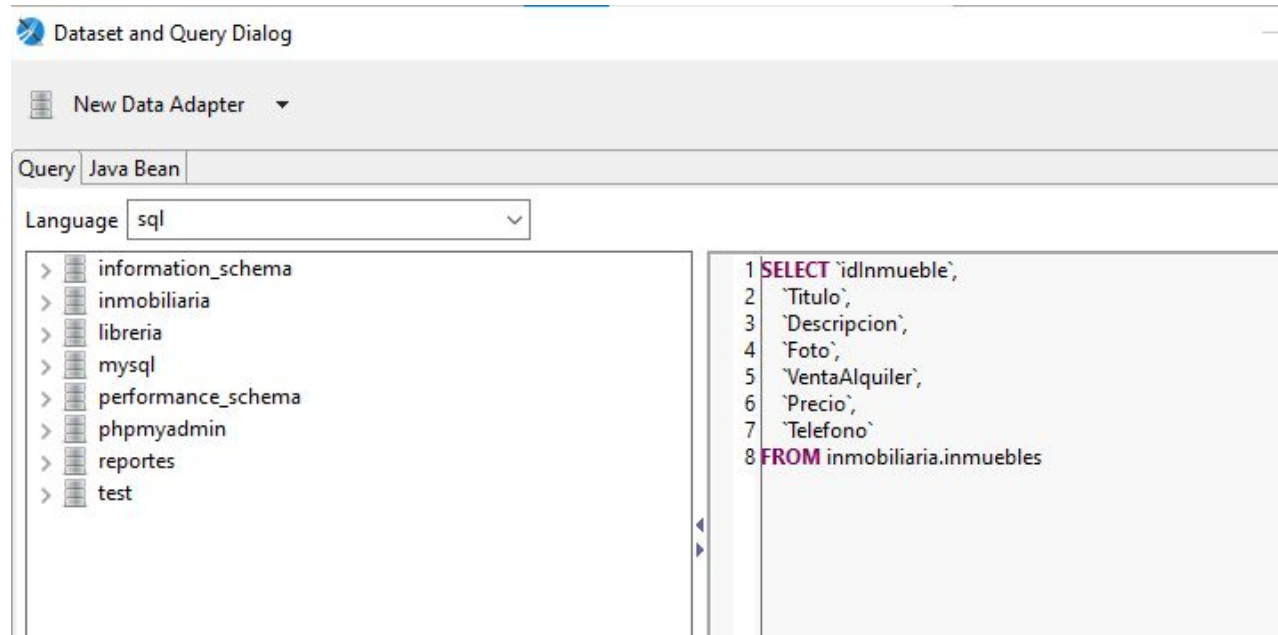


JasperReport en Netbeans parte 2: Paso de parámetros

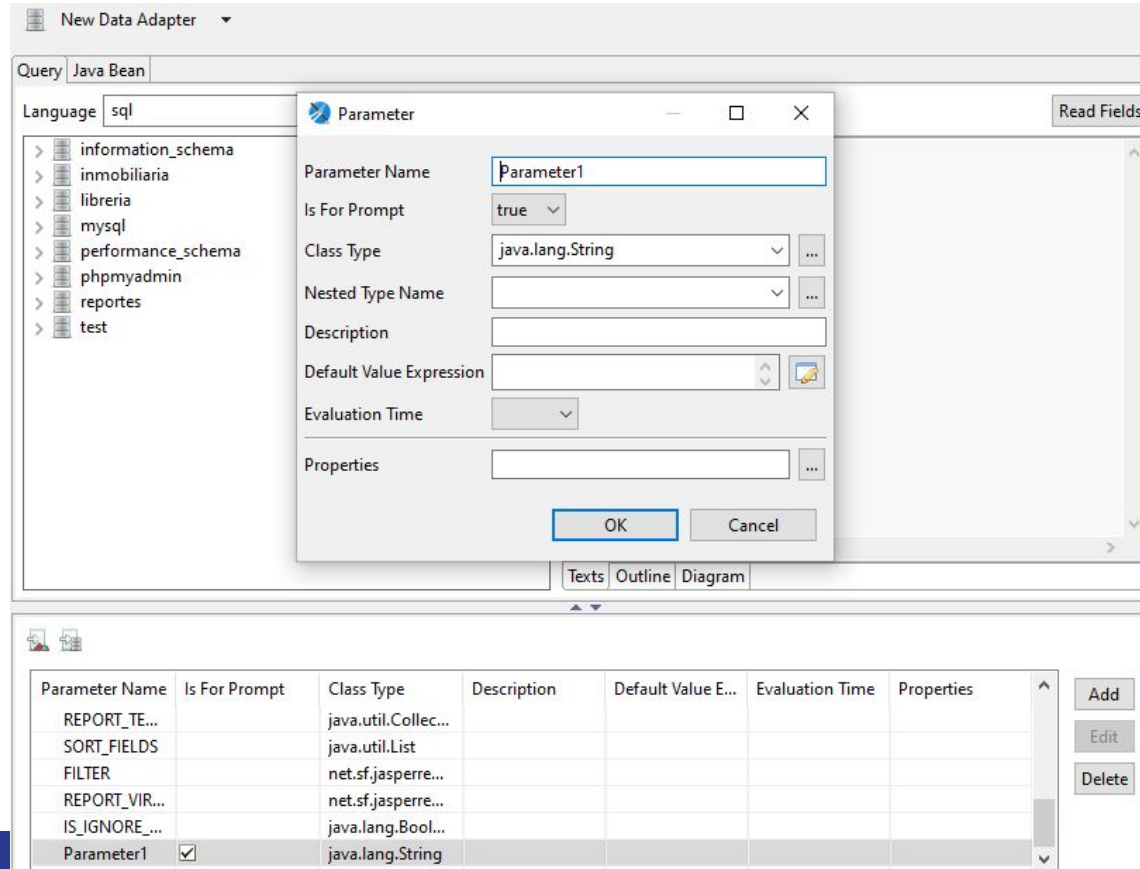
José Jarones Bueno

Añadir un parámetro a una consulta

En el apartado Data Adapter vamos a crear la consulta sql añadiendo un parámetro.



Pulsamos en la pestaña Parameter - Add



Creamos el parámetro

Parameter

Parameter Name:

Is For Prompt:

Class Type:

Nested Type Name:

Description:

Default Value Expression:

Evaluation Time:

Properties:

OK Cancel

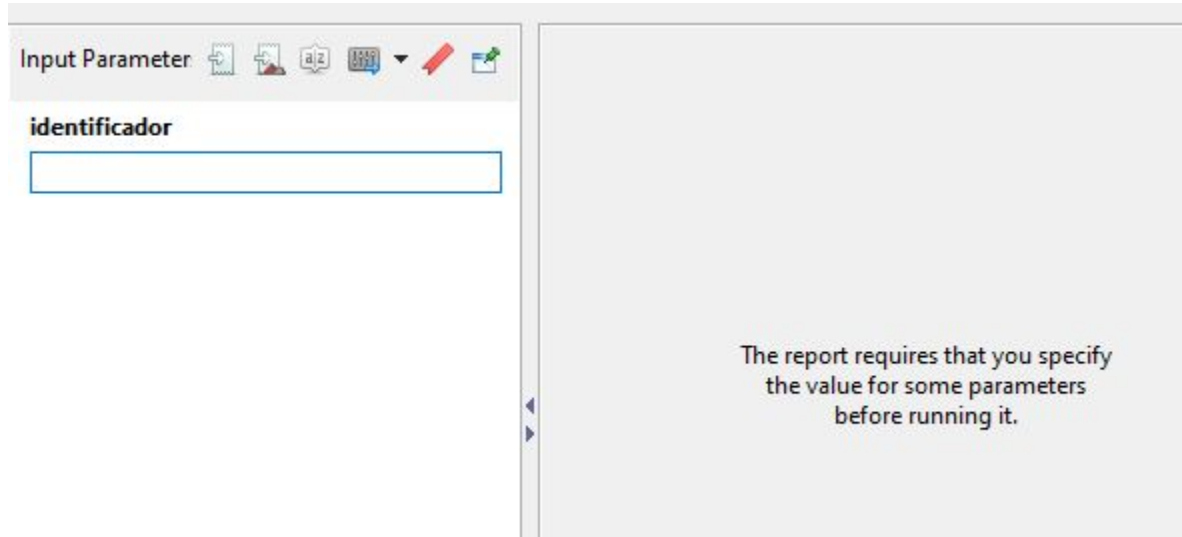
Lo añadimos a nuestra consulta

Podemos arrastrarlo

Los parámetros se reconocen
con la letra \$P delante.

```
1 SELECT `idInmueble`,  
2   `Titulo`,  
3   `Descripcion`,  
4   `Foto`,  
5   `VentaAlquiler`,  
6   `Precio`,  
7   `Telefono`,  
8   venta  
9 FROM inmobiliaria.inmuebles  
10 where idInmueble= $P{identificador}
```

Al pulsar sobre preview nos pedirá introducir el parámetro

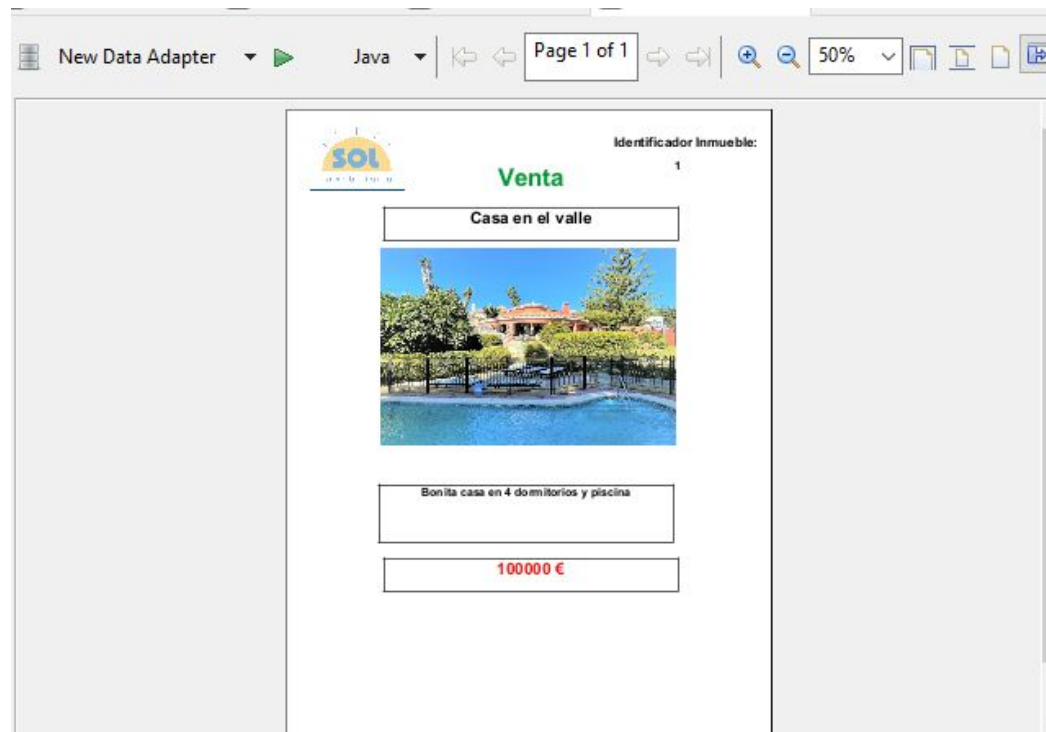


Input Parameter

identificador

The report requires that you specify the value for some parameters before running it.

Ejemplo



En Netbeans

Antes de realizar el Jasperprin, debemos pasar los parámetros necesarios para el informe. (Si no los pasamos, no funcionará).

```
Map parametros = new HashMap();  
int id=1;  
parametros.put("identificador", id);  
//Cuarto, generar el prin - (enlazar el fichero compilado, con la conexión).  
JasperPrint prin = JasperFillManager.fillReport(archivo, parametros, cnx);  
//Quinto , Generar el PDF con el prin anterior  
JasperExportManager.exportReportToPdfFile(prin, "Inmueble.pdf");
```

En
JasperPrint
paso los
parámetros


```
try{
//Primero, abrir el fichero (FILE, el jrxml)
File file = new File(getClass().getClassLoader().getResource("Inmueble.jrxml").getFile());
//Segundo, compilar el jrxml
JasperReport archivo = JasperCompileManager.compileReport(file.getAbsolutePath());
//Tercero, crear la conexión a la bbdd
Connection cnx;
Class.forName("com.mysql.jdbc.Driver");
cnx= DriverManager.getConnection("jdbc:mysql://localhost:3307/inmobiliaria?user=root&password=");
Map parametros = new HashMap();
int id=1;
parametros.put("identificador", id);
//Cuarto, generar el prin - (enlazar el fichero compilado, con la conexión).
JasperPrint prin = JasperFillManager.fillReport(archivo, parametros, cnx);
//Quinto , Generar el PDF con el prin anterior
JasperExportManager.exportReportToPdfFile(prin, "Inmueble.pdf");
}catch(Exception e){
    e.printStackTrace();
}
```

Sentencias IF

En los campos se puede llamar a funciones y generar diferentes sentencias, por ejemplo IF(condicion, valor si true, valor si false)

Expression Editor

Edit the JasperReports expression that returns the expected type.

The screenshot shows the Expression Editor window with the expression `IF($F{venta}, "Venta", "Alquiler")` entered in the top text field. Below the text field is a list of built-in functions. The **IF** function is selected, and its details are shown on the right. The details include the return type `java.lang.Object` and the description: "Returns one of two values, depending on a test condition. This function returns an object of type: `java.lang.Object`". The configuration fields are as follows:

Field	Value
Test condition *	<code>\$F{venta}</code>
Value 1 (true) *	<code>"Venta"</code>
Value 2 (false) *	<code>"Alquiler"</code>

At the bottom of the window, there is a checkbox for "Value Class (backward compatibility only)" which is currently unchecked. The bottom right corner contains "Finish" and "Cancel" buttons.