Advanced Form Data Binder

- Introduction
- Advanced Form Data Binder Properties
  - Configure Advanced Form Data Binder
  - Advanced
    - Filter
    - Aggregate Query
    - Expression Columns

Introduction

Advanced Form Row Data Binder is an extended version of the default Form Data Binder. It allows you to add in **Filter Conditions** in a guided and friendly manner.

Advanced Form Data Binder Properties

Configure Advanced Form Data Binder



Figure 1: Configure Advanced Form Data Binder

Name	Description
Form	Source form to retrieve data from.
Joins Form Data Table	**Name** / **Description** Form Data Table Name — Target table to join with Field — Target table field to join with Join Field Id — Parent field Id to join with

Sample

In the screenshot example in Figure 1, such configurations can be presented with the following SQL.

Sample SQL

```
SELECT * FROM "Claim Entry" entry JOIN hr_expense_claim claim ON claim.id = entry.claim
```

Advanced

Filter

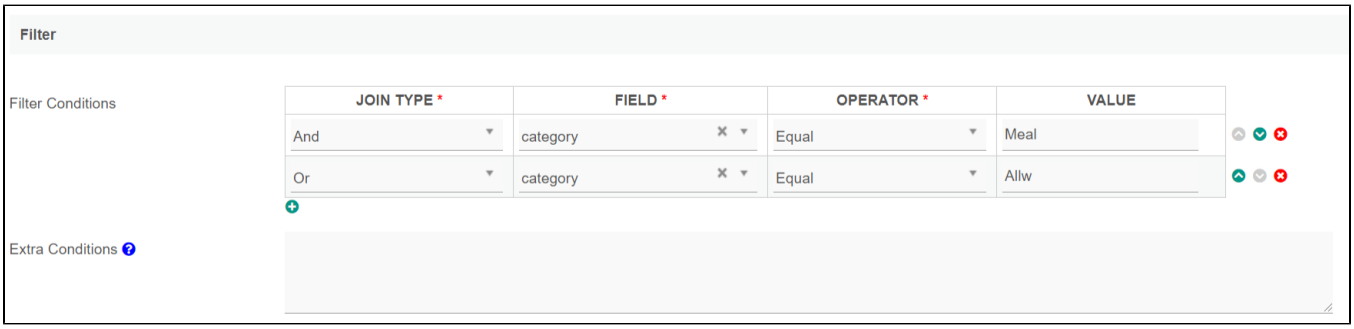
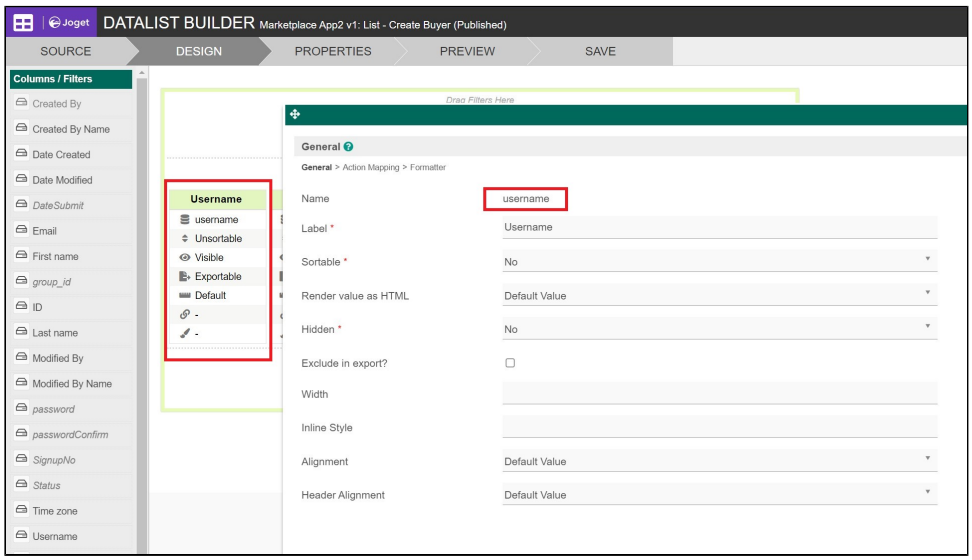


Figure 2: Advanced > Filter

Name	Description
Filter Conditions	Filter Conditions

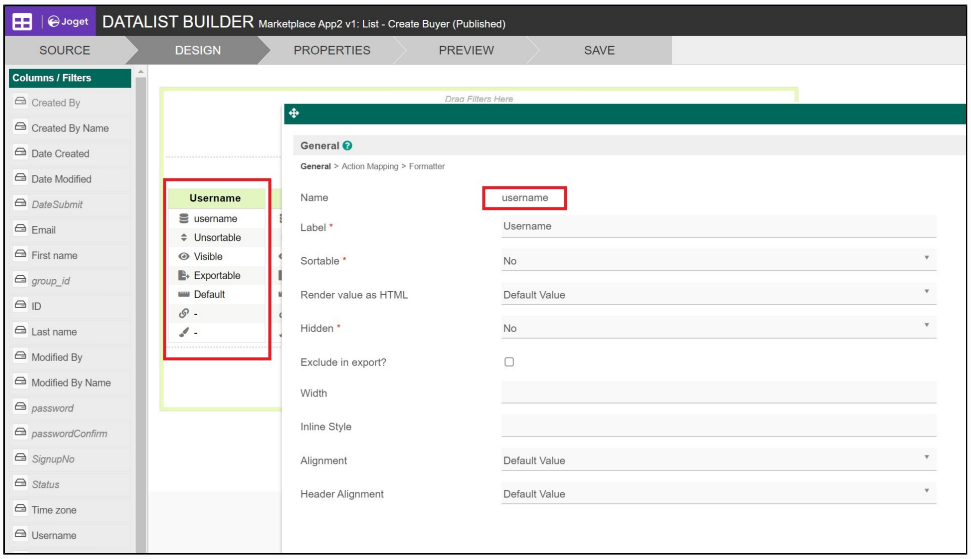
Name	Description
Join Type	• And • Or
Field	Field ID. (e.g. username) 
Operator	• Equal • Not Equal • Greater Than • Greater Than Or Equal • Less Than • Less Than Or Equal • Like • Not Like • In • Not In • Is True • Is False • Is Null • Is Not Null
Value	Filter value

Extra Conditions

Additional condition(s) for filtering the data set. HQL is expected here.

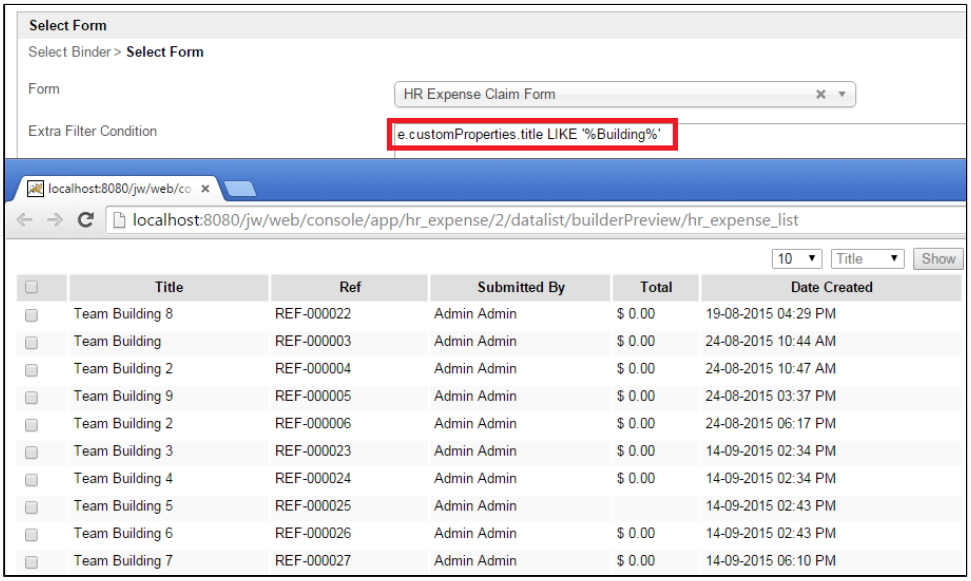
Syntax Query

Start your filter name with **e.customProperties.** followed by the field id (i.e. username)



HQL is accepted

You may even use an operator such as "LIKE" to narrow down your data set.



Sample

```
e.customProperties.title = 'Trip'
```

A hash variable is accepted here.

Sample

```
e.customProperties.submitted_by = '#currentUser.id#'
```

Userview Key can be used as part of the condition.

Sample

```
e.customProperties.category_id = '#userviewKey#'
```

Aggregate Query

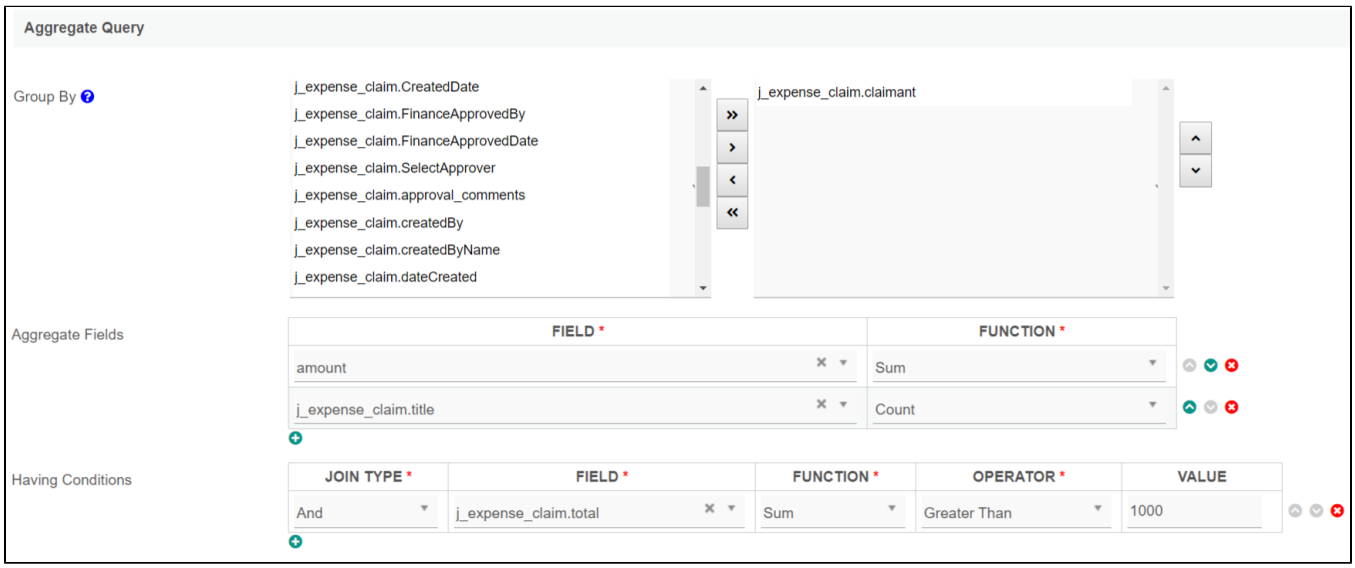


Figure 3: Advanced > Aggregate Query

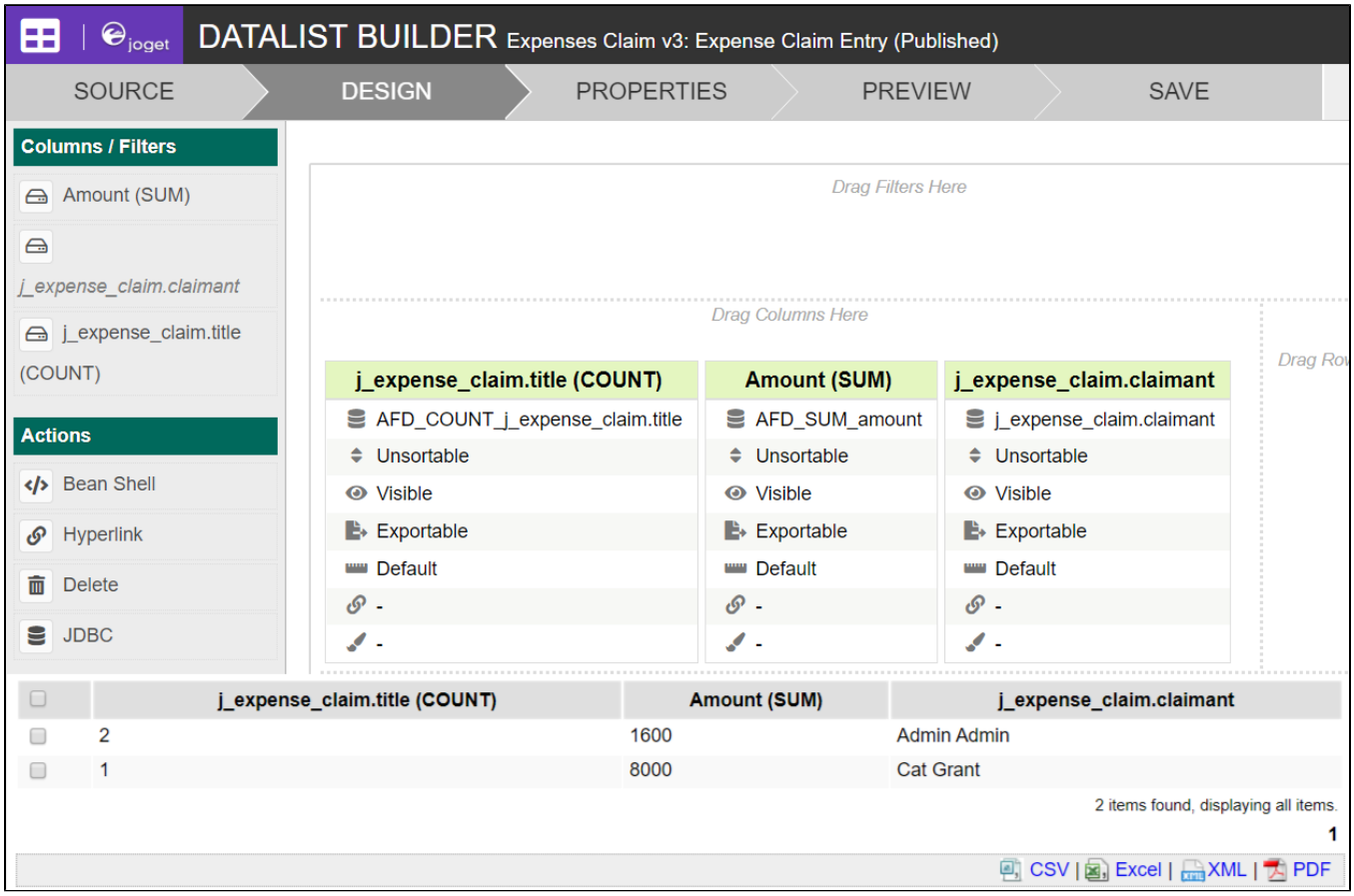


Figure 4: The configurations shown in **Figure 3** will produce the following sample result.

Name	Description
Group By	Add **grouping** clause/function to the eventual data set. This can be used together with **Aggregate Fields** above. In figure 3, the "amount" field will be summed up by "claimant", shown in per record row.
Aggregate Fields	This field will be displayed once any number of columns has been added into the **Group By** field. The select field is to aggregate. • Count • Count Distinct • Sum • Min • Max • Avg In the sample screenshot above, the "amount" field will be put into the "Sum" function, and "Count" will be applied to "title".
Having Conditions	This field will be displayed once any number of columns has been added into the **Group By** field. The **HAVING** clause enables you to specify conditions that filter which **group** results appear in the final results. The WHERE clause places conditions on the selected columns, whereas the **HAVING** clause places conditions on **groups** created **by** the **GROUP BY** clause. Read more at http://www.dofactory.com/sql/having

Expression Columns

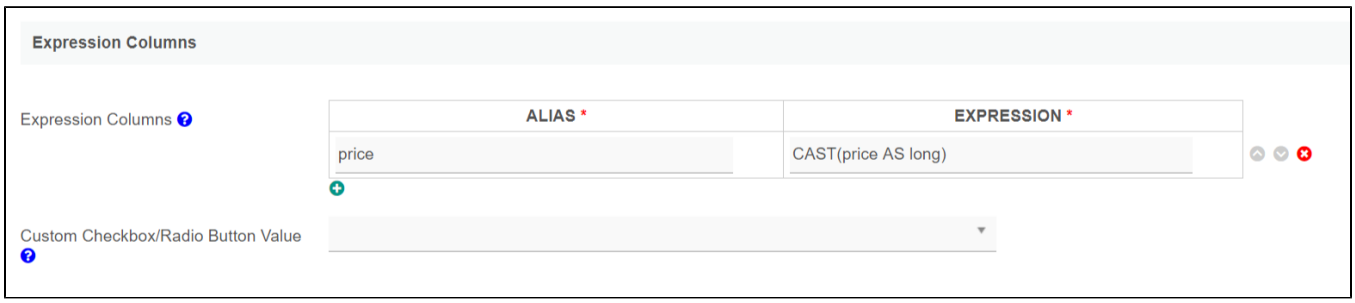


Figure 5: Advanced > Expression Columns

Name	Description

Expression Columns	An additional column can be added in this expression columns using Hibernate Query Language (HQL). This is especially useful when you need to perform additional computation on multiple columns.
	
	Example 1 - Cast column to data type "long".
	
	Expression
	`CAST(price AS long)`
	
	`SUBSTRING(CAST(dateCreated as string),1,10)  // To extract the date from the datetime database column`
	
	Expression
	`CAST(e.customProperties.sales_price AS long) - CAST(e.customProperties.price AS long)`
	
	Example 2 - Concatenate multiple columns into one.
	
	Expression
	`CONCAT(first_name, ' ', last_name)`
	
	Expression
	`first_name ||' '|| last_name`

Custom Checkbox/Radio Button Value	Define custom record ID to be used to pass over to column action. Defaulted to ID. 