

## BooleanFilter, How To

The aim of this How To is to explain in detail the BooleanFilter and the DefaultFilterValues behaviors.

### 1. The DefaultComponent definition

Suppose you have a DefaultComponent based on the table Customer.

Like this :

```
Customer
  id
  firstName
  lastName
  vat
```

To get data from this DefaultComponent, you should use the following code:

```
DefaultComponent customer=new Launcher("project")
.launch("Customer");
MatriceMap data=customer.getNewDataValues();
System.err.println(data);
```

This code produces the following SQL statement :

```
SELECT ID, FIRST_NAME, LAST_NAME, VAT FROM CUSTOMER
```

which results in a 2d matrix with 4 columns and n rows.

See the MatriceMap HowTo to study how to browse and manipulate data.

## 2. Adding a Filter

Should you need to add a filter to this request to obtain, for example, the customer who have the id 153, there are two ways to do this.

### a. *The stateful way*

The code :

```
DefaultComponent customer=new Launcher("project")
                                .launch("Customer");
customer.addAdditionalFilter("id",new Long(153));
MatriceMap data=customer.getNewDataValues();
System.err.println(data);
```

After the processing of the second row (addAdditionalFilter), every getNewDataValues() will return this row.

The following example is more declarative and produces the same result:

```
DefaultComponent customer=new Launcher("project")
                                .launch("Customer");
DefaultFilterValue df=new DefaultFilterValue();
df.setSourceFieldId(customer.getField("id").getId());
df.setValue(new Integer(153));
BooleanFilter bf=new BooleanFilter(df);
customer.setAdditionalFilter(bf);
MatriceMap data=customer.getNewDataValues();
System.err.println(data);
```

To remove this additionalFilter, please use le following command :

```
customer.removeAdditionalFilter();
// or for older versions of Junivers
customer.setAdditionalFilter(null);
```

## ***b. The stateless way***

The code :

```
DefaultComponent customer=new Launcher("project")
                                .launch("Customer");
DefaultFilterValue df=new DefaultFilterValue();
df.setSourceFieldId(customer.getField("id").getId());
df.setValue(new Integer(153));
BooleanFilter bf=new BooleanFilter(df);
MatriceMap data=customer.getNewDataValues(bf);
System.err.println(data);
```

In this example, the state of DefaultComponent is not modified; a call to getNewDataValues() will return all rows.

Both ways produce the following SQL statement :

```
SELECT ID, FIRST_NAME, LAST_NAME, VAT FROM CUSTOMER WHERE ID=153
```

### 3. BooleanFilter (combinaisons of filters)

This section describes how to combine filters with boolean clauses.

#### a. *Combinaison of two clauses*

Should you produce the following SQL statement,

```
SELECT ID, FIRST_NAME, LAST_NAME, VAT FROM CUSTOMER
WHERE FIRST_NAME='Marcel' AND LAST_NAME='DUCHEMIN'
```

you have to construct the following BooleanFilter :

```
DefaultFilterValue ff=new DefaultFilterValue();
ff.setSourceFieldId(customer.getField("firstName").getId());
ff.setValue("Marcel");

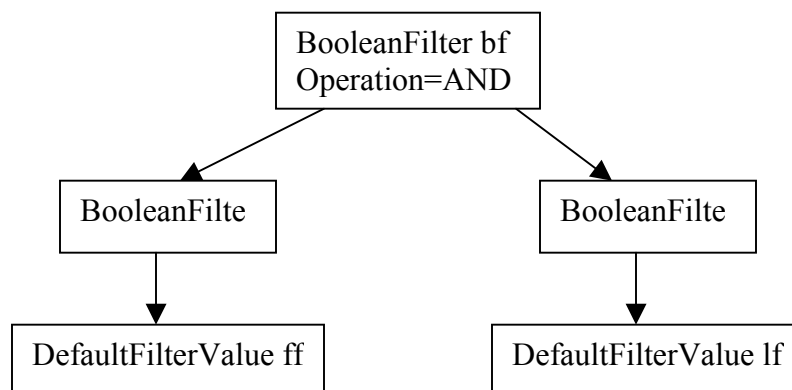
DefaultFilterValue lf=new DefaultFilterValue();
lf.setSourceFieldId(customer.getField("lastName").getId());
lf.setValue("DUCHEMIN");

BooleanFilter bf=new BooleanFilter(BooleanFilter.BOOL_AND);
bf.addChildBooleanFilter(ff);
bf.addChildBooleanFilter(lf);
```

**A Filter is a tree in which each node is a BooleanFilter.**

**A BooleanFilter could be a parent of other Boolean filters OR contains a DefaultFilterValue.**

There is a graphical view of the previous example :



## ***b. Trees of clauses***

Should you produce the following SQL statement,

```
SELECT ID, FIRST_NAME, LAST_NAME, VAT FROM CUSTOMER
WHERE FIRST_NAME='Marcel' AND (LAST_NAME='DUCHEMIN' OR
LAST_NAME='LAGRANGE')
```

please construct the following BooleanFilter :

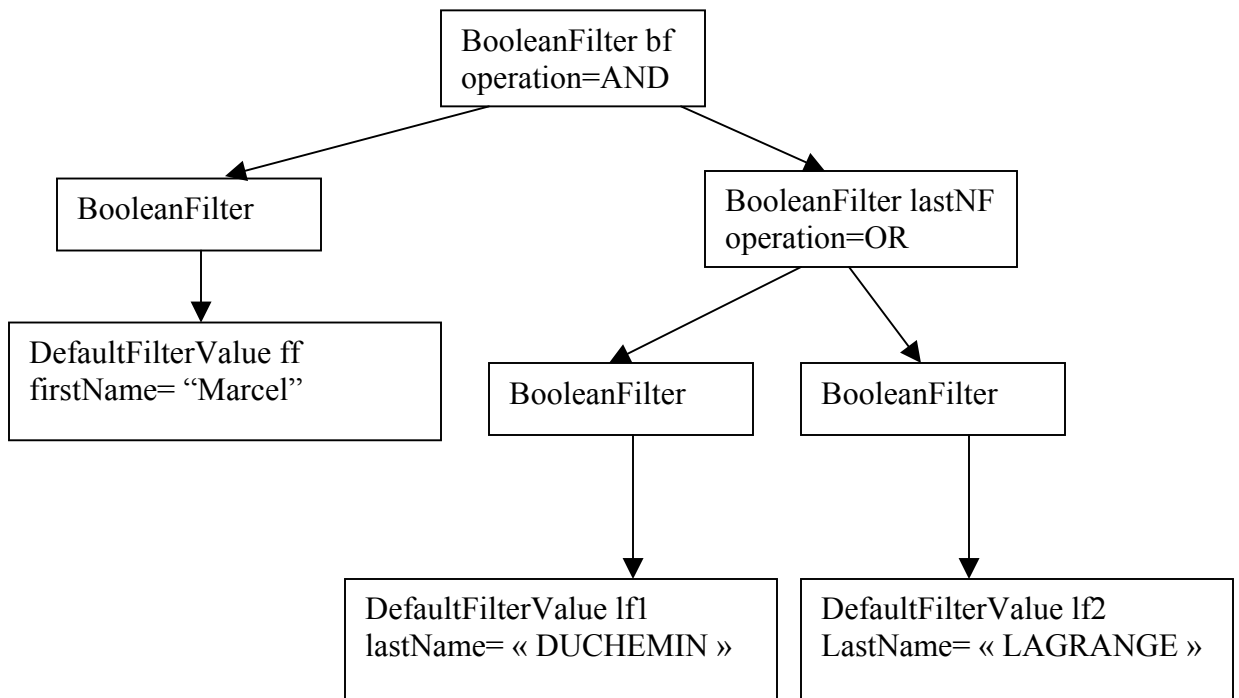
```
DefaultFilterValue ff=new DefaultFilterValue();
ff.setSourceFieldId(customer.getField("firstName").getId());
ff.setValue("Marcel");

DefaultFilterValue lf1=new DefaultFilterValue();
lf1.setSourceFieldId(customer.getField("firstName").getId());
lf1.setValue("DUCHEMIN");

DefaultFilterValue lf2=new DefaultFilterValue();
lf2.setSourceFieldId(customer.getField("firstName").getId());
lf2.setValue("LAGRANGE");

BooleanFilter rootF=new BooleanFilter(BooleanFilter.BOOL_AND);
BooleanFilter lastNF=new BooleanFilter(BooleanFilter.BOOL_OR);
rootF.addChildBooleanFilter(ff);
rootF.addChildBooleanFilter(lastNF);
lastNF.addChildBooleanFilter(lf1);
lastNF.addChildBooleanFilter(lf2);
```

Here is a graphical view of the previous example :



## 4. Special filter values

### a. Case insensitive

Should you need case insensitive filters,  
Please use the following method :

```
df.setIgnoreCase(true);
```

The request will be case insensitive and will ignore accents.  
The WHERE clause of the select statement will look like this :

```
WHERE UPPER(CONVERT(FIRST_NAME,'US7ASCII')) =  
UPPER(CONVERT('Marcel','US7ASCII'))
```

## ***b. Non-equals filters***

The DefaultFilterValue defines the following comparison operations :

```
public static final int COMP_CUSTOM=-1;
public static final int COMP_EQUAL=1;
public static final int COMP_DIFFERENT=2;
public static final int COMP_GREATER=3;
public static final int COMP_LOWER=4;
public static final int COMP_GREATER_EQUAL=5;
public static final int COMP_LOWER_EQUAL=6;
public static final int COMP_STARTWITH=7;
public static final int COMP_ENDWITH=8;
public static final int COMP_CONTAIN=9;
```

Should you need to filter with clauses like greater, lower, ... please use the following system:

```
DefaultFilterValue df=new DefaultFilterValue();
df.setSourceFieldId(customer.getField("firstName").getId());
df.setValue("A");
df.setCompOp(DefaultFilterValue.STARTWITH);
```

SQL generated :

```
SELECT ID, FIRST_NAME, LAST_NAME, VAT FROM CUSTOMER
WHERE FIRST_NAME LIKE 'A%'
```

### c. DataSource specific filters (RDBMS Where clauses)

In case of very complicated search process, you can use by example manual generated SQL where clause.

It is not recommended to use such a solution because it is not cross datasource compatible: a string “Where clause” could not be applied on a LDAP data source. So be careful and try first to implement the search process with Juniver filters.

Code example :

```
DefaultFilterValue df=new DefaultFilterValue();
df.setSourceFieldId(customer.getField("id").getId());
df.setValue(new Integer(153));
df.setCompOp(DefaultFilterValue.COMP_GREATER);
df.setCustomOperation("%SOURCE% IN(SELECT ID FROM SUPER_CUSTOMER
WHERE ID=%TARGET%)");
```

SQL generated (in bold) :

```
SELECT ID, FIRST_NAME, LAST_NAME, VAT FROM CUSTOMER
WHERE FIRST_NAME='Marcel' AND (LAST_NAME='DUCHEMIN' OR
LAST_NAME='LAGRANGE') AND (CUSTOMER.ID IN(SELECT ID FROM
SUPER_CUSTOMER WHERE ID=153))
```

## 5. Warning

Don't forget there are three levels of filters :

- 1) Definition – level
- 2) LinkFilter & AccessRights
- 3) Additional filters

Each levels are used as children of an “And” filter.

If you use AdditionalFilters on components, the current filter of the component is still used.

The stateless way (getNewDataValues(bf)) bypass the points 2 and 3.

If you want to use the stateless-way combined with the three previous filters, please use the following code :

```
DefaultComponent customer=new Launcher("project")
                                .launch("Customer");
DefaultFilterValue df=new DefaultFilterValue();
df.setSourceFieldId(customer.getField("id").getId());
df.setValue(new Integer(153));
BooleanFilter bf=new BooleanFilter(BooleanFilter.BOOL_AND);
bf.addChildBooleanFilter(customer.getCurrentFilter());
bf.addChildBooleanFilter(df);
MatriceMap data=customer.getNewDataValues(bf);
System.err.println(data);
```

## 6. Tips

If a component has selected rows, you can obtain a BooleanFilter of those selected rows like this :

```
BooleanFilter bf=customer.getSelectionFilter();
```

This filter uses primary keys of the component to generate the filter.

After inserting data, you can get the filter of the inserted data. Like this :

```
BooleanFilter bf=customer.processFieldsInsertion();
```

You can use this to reselect a row after an insertion :

```
customer.setSelectedRows(customer.getNewDataValues(  
customer.processFieldsInsertion()));
```

FiltersValues doesn't need to be filters of children of the component.  
You could have the Component ...

```
Customer  
    id  
    firstName  
    lastName
```

... and filter the request on VAT AND (adresse OR openCommands) AND ...  
The fields just need to be children of the sources of the component.

## 7. Conclusion

BooleanFilters first seem very complex, but after utilisation you'll realize it is the  
simplest way to modelize filters.