Fine-tuning the Products/Services autocomplete

`finetuning-product-service-autocomplete-corebos.md`

Fine-tuning the Products/Services autocomplete

There has been an update that will allow you to fine-tune the autocomplete for products and services. Let's dive in and explain how you can set the options:

First off: selecting the fields the autocomplete searches in

By default, typing something in the product lines will fire a search in the following fields by default:

Products

- productname
- manufacturer part no
- vendor part no

Services

- servicename

Obviously, you might want to change this behaviour. Let's first discuss how you can change this per module (the ones where the search typically lives: Quotes, SalesOrders, Invoices and PurchaseOrders).

Create a business map of type 'FieldInfo'

It is already possible to create a BusinessMap to set certain fields as 'autocomplete', as you can read here. You will see that in order to load this BusinessMap on the module you desire, you need to *name* the BusinessMap 'MODULENAME_FieldInfo' and create the structure as defined in the wiki (see the link). What you can do is extend this map (if you already have one) or create the map with a special fieldname, that does not really exist: 'cbProductServiceField'. This will be ignored (unless in the very odd case where you would actually have a field that has that name) during the rest of the BusinessMap handling, except for the Product/Service autocomplete.

Now let's look at how a BusinessMap with only that field might look like:

```
<map>
    <originmodule>
        <originname>Quotes</originname>
    </originmodule>
    <fields>
        <field>
            <fieldname>cbProductServiceField</fieldname>
            <features>
                <feature>
                    <name>searchfields</name>
                    <values>
                        <value>
                            <module>Products</module>
                            <value>productname,mfr_part_no,vendor_part_no,
                        </value>
                        <value>
                            <module>Service</module>
                            <value>servicename,service_no</value>
                        </value>
                    </values>
                </feature>
            </features>
        </field>
    </fields>
</map>
```

Here, you can see the 'normal' autocomplete directives like 'searchfields' have been used as much as possible to make sure the map-validation logic stays intact. The searchfield directive can receive two values (you *can* choose to only use one, the other one will then use the defaults): "Products" and "Service" (**do** mind the capitalization).

So let's break it down a bit more:

```
<values>
    <value>
        <module>Products</module>
        <value>productname,mfr_part_no,vendor_part_no,product_no</value>
```

```
    </value>
    <value>
        <module>Service</module>
        <value>servicename,service_no</value>
    </value>
</values>
```

Is the bit where you create a 'value' directive for each of the two modules that are being searched. Each 'value' receives a 'module' with the appropriate name (again mind the capitalization) and a comma-separated list in its 'value' directive. **Make sure you use columnnames here**, as there will be no fieldname-to-columnname conversion attempted.

Just to make it very clear: this will alter the behaviour for **all** users, but **only** for the module you create the BusinessMap for.

Search in compounded fields

Lets say you've created a field in the Products module called 'brand' where you store the brand of the product, while in the productname field, you store the model. So for instance a TV could have 'Sony' in the brand field, while having 'KDL400 EX' in the productname field. Now what if you wanted to let users search in a combination of the brand and model fields. So you'd want to search on 'Sony KDL400 EX' and find the one you're looking for. Well, since november 2020, you can do that. You can create a 'compounded' searchfield, meaning you can create a searchfield that is a contraction of one or more fields. The syntax there is: use square brackets to surround the fields you want to compound, and use a pipe character to separate them. So let's say you have a 'brand' field that's called `cf_543` (custom fields are supported), you would use `[cf_543|productname]`.

You can use these compounded fields in conjunction with normal searchfields, so if we were to expand the previous example, it would look like this:

```
<values>
    <value>
        <module>Products</module>
        <value>[cf_543|productname],productname,mfr_part_no,vendor_part_no
    </value>
    <value>
        <module>Service</module>
        <value>servicename,service_no</value>
    </value>
</values>
```

Known limitations of the searchfields

There are two major things you need to take into account when setting up searchfields, that mainly have to do with the difference between the database and the values you see on screen:

- Translation of picklist values will not happen. So, if you use picklists that are installed with the application or on modules or updates, those usually use English values in the database but provide localized values on screen. The autocomplete will always search on the database and won't take the translation into account. So basically: create your own picklist entries and use those if searching based on picklist values is important to you.
- Reference fields will not be searchable as well. So if, like in the example above, you want to use the vendor as the 'brand', that won't work. What you would do in that case is create a textfield that carries the vendorname so the autocomplete can search in that field instead.

Secondly: Constrain the search based on criteria you can choose

Now, you've altered the behaviour of the fields that are used to search in, but maybe you have a more delicate use-case. Imagine you have a group of users, grouped by their role, that don't need to be able to search through all of the products of services you have in your database.

A use-case could be that you have products in your database that are sold as complete units through quotes by salespeople, but also have a lot of small parts for those units as products in the database. A far as the system in concerned: they're both just products. Searching for 'Product A' might bring up two results:

- The product with a name of 'Product A'
- A small spare part that is called 'Spare bolt for Product A'

You don't want that second part to show up in the search results in Quotes, since they never will be Quoted and just clutter up the search results. Let's see how we can do that, again using the regular directives in the BusinessMap. Imagine you had a checkbox in Products that was called 'commercial' and 'Product A' was checked, but 'Spare bolt for Product A' was not.

Expand the map from before

Let's use our previous map and extend it:

```
<map>
    <originmodule>
        <originname>Quotes</originname>
```

```
    </originmodule>
    <fields>
        <field>
            <fieldname>cbProductServiceField</fieldname>
            <features>
                <feature>
                    <name>searchfields</name>
                    <values>
                        <value>
                            <module>Products</module>
                            <value>productname,mfr_part_no,vendor_part_no,
                        </value>
                        <value>
                            <module>Service</module>
                            <value>servicename,service_no</value>
                        </value>
                    </values>
                </feature>
                <feature>
                    <name>searchcondition</name>
                    <value>{"Products" : [
                                    {
                                    "field" : "commercial",
                                    "operator" : "equals",
                                    "value" : "1"
                                    },
                                    "OR",
                                    {
                                    "field" : "registable",
                                    "operator" : "equals",
                                    "value" : "1"
                                    }
                                ],
                            "Service" : [
                                    {
                                    "field" : "qty_per_unit",
                                    "operator" : "smaller",
                                    "value" : "2"
                                    }
                                ]
                            }
                    </value>
                </feature>
            </features>
        </field>
    </fields>
</map>
```

Look at the 'features' in the 'field' directive. It has been extended with a new feature that is standard for the FieldInfo Business Map called 'searchcondition'. Normally, that value would specify *how* you want to search ('startswith', 'contains', etc.). In the Products/Services autocomplete we always use 'contains', that is hard-coded. In stead, in the special 'cbProductServiceField' field directive, we use it for something else.

The 'seachcondition' directive in this context is being used to constrain the search results and receives a JSON object:

```
{"Products" : [
        {
        "field" : "commercial",
        "operator" : "equals",
        "value" : "1"
        },
        "OR",
        {
        "field" : "registable",
        "operator" : "equals",
        "value" : "1"
        }
    ],
"Service" : [
        {
        "field" : "qty_per_unit",
        "operator" : "smaller",
        "value" : "2"
        }
    ]
}
```

The JSON object has two entries: "Products" and "Service", each of those will receive an array. The entries of that array should follow an odd/even paradigm:

The odd entries should receive a restraint

Every odd entry should be a JSON object like this (taken from the JSON above):

```
{
"field" : "commercial",
"operator" : "equals",
"value" : "1"
}
```

with three keys: field, operator and value.

Beware here, like before: the field should contain the **columnname** you want to search, no field-to-columnname will be attempted. The keyname is chosen to keep consistency with the rest of the application.

The operator has three options:

- **equals**
- **smaller**
- **greater**

which all behave like you would expect.

The even entries should be the glue

You can glue more than one restrain together by using 'OR' or 'AND' operators. So the above JSON object would result in:

```
(commercial = 1 OR registable = 1)
```

for Products and

```
(qty_per_unit < 2)
```

for Services.

Again: **this map will work for all users but only for the module you specify in the map name**

Third: selecting which fields should be filled with which information

A small disclaimer on this functionality: at the time of writing, this will only work on the 'description' field.

As with 'normal' autocomplete fields, you may want to select what happens when a result from the list of suggestions is selected. In the regular autocomplete map, the 'fillfields' directive is used to do that. Observe this excerp:

```
<feature>
    <name>fillfields</name>
    <values>
        <value>
            <module>Accounts</module>
            <value>forecast_amount=annual_revenue</value>
        </value>
```

```
            <value>
                <module>Contacts</module>
                <value>leadsource=leadsource,assigned_user_id=assigned_user_id
            </value>
        </values>
    </feature>
```

** keep in mind this is **part** of a field directive*

As you can see, you provide the source modules you want the information to come from and state a comma-separated list of key/value pairs in the 'value' directive. Now let's take a look at how this implementation works when you use it in the context of the Products/Service search:

```
    <map>
        <originmodule>
            <originname>Quotes</originname>
        </originmodule>
        <fields>
            <field>
                <fieldname>cbProductServiceField</fieldname>
                <features>
                    <feature>
                        <name>searchfields</name>
                        <values>
                            <value>
                                <module>Products</module>
                                <value>productname,mfr_part_no,vendor_part_no,
                            </value>
                            <value>
                                <module>Service</module>
                                <value>servicename,service_no</value>
                            </value>
                        </values>
                    </feature>
                    <feature>
                        <name>searchcondition</name>
                        <value>{"Products" : [
                                        {
                                        "field" : "commercial",
                                        "operator" : "equals",
                                        "value" : "1"
                                        }
                                    ],
                                "Service" : [
                                        {
                                        "field" : "qty_per_unit",
                                        "operator" : "smaller",
                                        "value" : "2"
                                        }
                                    ]
```

```
                        }
                    </value>
                </feature>
                <feature>
                    <name>fillfields</name>
                    <values>
                        <value>
                            <module>Products</module>
                            <value>description=''</value>
                        </value>
                        <value>
                            <module>Service</module>
                            <value>description=description</value>
                        </value>
                    </values>
                </feature>
            </features>
        </field>
    </fields>
</map>
```

This is the entire map from the previous examples, expanded with the 'fillfields' feature. In contrary to regular 'FieldInfo' BusinessMaps, you are only allowed the 'Products' and 'Service' module. Any other module name will simply be ignored. As stated before, at the moment you can only select the fill-behaviour of one field: description (the product or service description below the name that is filled when a selection is made from the suggestion list).

There are a couple of options available, of which 'description=description' is the default one, and is how the search will behave when no 'fillfields' directive is stated. Other options include:

- '' (empty string): this will effectively remove the auto-fill of the description. Of course it will respect the BusinessMap name, so a 'Quotes_FieldInfo' with this directive will only do its magic when searching in quotes.
- `columname` : So for instance 'description=servicecategory' will fill the description with the Service's category when selecting an entry from the suggestions. Custom field columnnames will automatically select the custom fields table.

** Again: keep in mind no fieldname->columnname will be attempted, so make sure to use columnnames*

Linking the maps to certain roles

Now for the fun part. Let's take the example we used before: 'Product A' should appear in Quotes when searching, 'Spare bolt for Product A' should not. What if we *did* want 'Spare bolt for Product A' to show up, but **only** when certain roles were searching? Let's say you want your salespeople to only find 'Product A' when searching in a Quote, but you want someone in the backoffice to be able to find **only** 'Spare bolt for Product A' when searching in a Quote? Well, you can do that!

Take the businessmap we had before, copy and rename it to something that explains its purpose, like 'Productsearch behaviour for salespeople'. Extend or modify it so it searches in the fields your salespeople will search in and more importantly, create some restraints like explained above (so for instance restrain to 'commercial' being equal to 1).

Now open the picklist editor and select the GlobalVariables module, and then the type picklist. Add a new type definition called `BusinessMapping_FieldInfo`. Go to the GlobalVariables module and create a new record. Select the `BusinessMapping_FieldInfo` type you just created and assign it to a role (*important*, your role needs to live above the role you want to assign to, or it won't be visible for selection!).

Now open the BusinessMap selection field and select the BusinessMap you just created ('Productsearch behaviour for salespeople' in our example). You've now told the application that *in stead* of looking for a default BusinessMap (by the name of for instance 'Quotes_FieldInfo'), you want it to use the Map you just selected for users that meet that role.

Now to tell it on which modules you want that to happen. Select the 'on modules' checkbox to tell it you want the module filtering to occur and select the modules you want this to happen on from the multiselect called 'Modules'. Let's for the sake of this example say you select Quotes and SalesOrders

What you've done now is tell the application: "When someone from the salespeople role searches a product of service in either a quote or salesorder, use this mapping to specify which fields they can search in, and which restraints the search should use".

The order of importance

So let's recap: Let's say you have a BusinessMap called 'Quotes_FieldInfo' in which you determine some search behaviour. You also create the second map with the GlobalVariable as explained above, what will happen on any search is:

- Is there a BusinessMap assigned to this role and does it want to work on this module? If yes, it is used
    - If not: Is there a 'general' BusinessMap for this module (for instance 'Quotes_FieldInfo')? If yes, it is used
        - None of the above? Resort to the defaults.

Last: setting the no. of results

There is a new Global Variable called `Application_ProductService_Search_Autocomplete_Limit` that will allow you to fine-tune the no. of results the Product/Service autocomplete will show. This will default to 10.