



## Best Practices Lotus Domino - @Formulas I

### @DBLookup - @DBCColumn

Your fast and inconvenient access to data residing in Notes Databases



13.04.2008

### Initial situation:

Lotus Domino provides a wonderful technology regarding keyword-fields: Selecting possible values based on formulas.

In general a developer will do this by writing a @DBCColumn or @DBLookup-Command in the selection-area of the keyword-field. So we are able to display selection options based of a set of given values or the content of other fields.

Now we assume that the 2 values in the view "va\_values" are "vegetables" and "fruits".

Now we can create a second field to enter additional information like the kind of vegetables. Here normally we use a @DBLookup to get information from another view based on the entered value.

Assuming that the view "va\_values" is showing possible vegetables and fruits in the second column we can now enable the user to enter the right values.

### Impact of this approach:

Having a closer look at the code displayed on the right handside there are a couple of potentially pretty negative impacts:

- No error handling has been included in the code – so if the view is not available we will get an ugly error message from the system.
- A lot of network activity and performance is wasted due to the multiple queries to the view "va\_values" unnecessarily.

Ad a)

Some small changes can make the code more stable regarding exceptions:

```
_A:=@DBLookup("": "NoCache";@DBName;"va_values";fd_kw_Category;2);  
_B:=@If(@IsError(_A); "No values found for your selection";@Trim(@Unique(_A)));  
_B
```

*Errorhandling in @Formula language:*

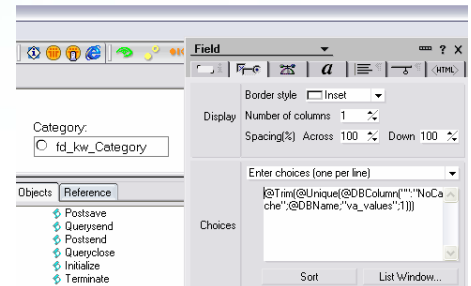
With an error-handling like above the database / form is still working – even if the @DBCColumn is causing an error.

Ad b)

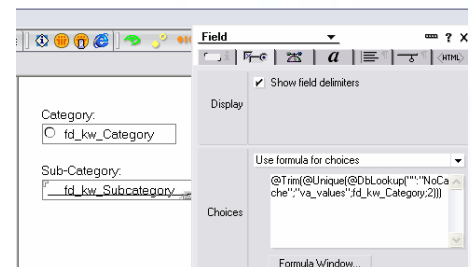
Important for the user satisfaction is the experienced speed of the system itself. Each query will cause delays and slow down the system.

In the situation above there are multiple queries depending from the user interaction: As you can see from the graphic on the right handside we will perform 3 queries if the user is selecting the wrong option the first time.

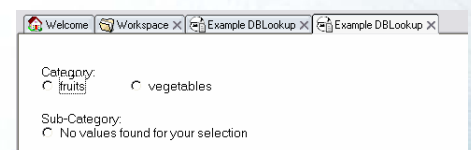
So what can we do to enhance the performance of the application?



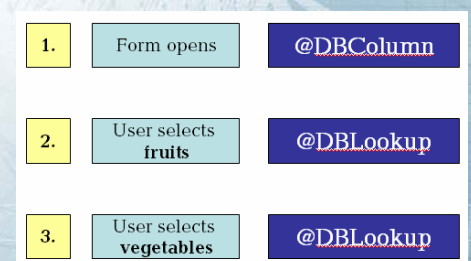
*Typical formula for binding data to keyword-fields*



*Typical way of binding hierarchical data to keyword-fields*



*Errorhandling avoids failing @DBLookup because of missing selection in a new document*



*Call tree of example*



# Best Practices Lotus Domino - @Formulas I

## @DBLookup - @DBCColumn

Your fast and inconvenient access to data residing in Notes Databases



13.04.2008

The answer is pretty simple: We have to **reduce** the number of queries (which are in general time consuming).

We will create a new field computed for display called "fd\_DSPValues".

This field will get its values from a @DBLookup to a "modified" view. Once the data is stored in the field, we can easily point our keyword-fields to this new field. Instead of asking a database on the server for values we will now ask a field in our Notes-UI and we are no more hurt by bad network responses or a user switching between both options a hundred of times.

If we compare "Ex. 1" with "Ex. 2" we can easily find out the differences. In "Ex. 1" we have a categorised view where we will get the values in the second column based on the entry shown in the first column.

In "Ex. 2" we will do only 1 @DBLookup and take the form name as the selection key. Now all values that are relevant for this form can be retrieved from the second column labeled by the "field name".

The following code can be used in the field "fd\_kw\_Category" to get the values for the category-field (equivalent to our @DBCColumn).

```
_KEY:="Category";
_A:=fd_DSPValues;
_B:=@Right(_A;_KEY + ":=");
_C:=@Trim(@Unique(_B));
_D:=@If(@IsError(_C);"No values found for your selection";@Trim(@Unique(_C)));
_D
```

Code for first field (fd\_kw\_Category):

And the code for the second field would be:

```
_KEY:=fd_kw_Category;
_A:=fd_DSPValues;
_B:=@Right(_A;_KEY + ":=");
_C:=@Trim(@Unique(_B));
_D:=@If(@IsError(_C);"No values found for your selection";@Trim(@Unique(_C)));
_D
```

Code for second field (fd\_kw\_SubCategory):

So the only difference is the value of the variable "\_KEY". While in the first field this is a "static" text like the field name – the other fields can use the values of the depending fields like we have shown with "fd\_kw\_Category".

Now we can really hide away the fields from our example and the user will get the identical functionalities but with a faster performance.

Cat	Sub-Category
▼ fruits	
	apple
	banana
	strawberry
▼ vegetables	
	carrot
	onion

Standard view for our example (Ex. 1)

Form to supply	Values
Test	Category:=fruits
Test	Category:=fruits
Test	Category:=fruits
Test	Category:=vegetables
Test	Category:=vegetables
Test	fruits:=apple
Test	fruits:=banana
Test	fruits:=strawberry
Test	vegetables:=carrot
Test	vegetables:=onion

Modified view fast solution (Ex. 2)

Field accessing « Ex. 2 »

Finished form with hidden elements visible



# Best Practices Lotus Domino - @Formulas I

## @DBLookup - @DBCColumn

Your fast and inconvenient access to data residing in Notes Databases



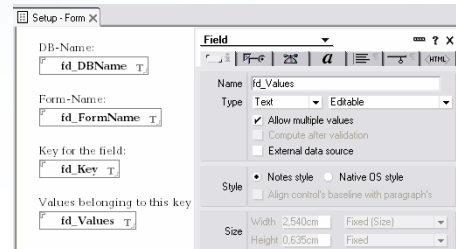
13.04.2008

### Cookbook @DBLookUp - @DBCColumn:

#### Step 1 :

For accessing data from keyword fields we should use ordinary Notes documents that can be shown in a view.

When creating these documents we should also give them an indicator of their target form as an additional value :

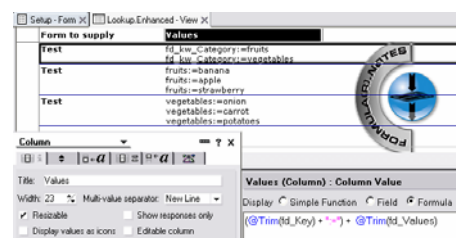


Form for setup values (Step 1)

#### Step 2 :

Next we have to create a view where we can access the data entered with the form described above. The best solution here is to sort the first column by the field « fd\_FormName » and have a second column displaying each value combined with its key.

If we tell the 2<sup>nd</sup> column to display different values as new entries, our @DBLookup will retrieve a list of all possible values.



Settings for 2<sup>nd</sup> column (Step 2)

```
(@Trim(fd_Key) + " ") + @Trim(fd_Values)
```

#### Code for Column "2"

#### Additional comment:

A system like that can easily be used to serve more forms and databases. We just have to include also the DB-Name in the first column for our initial query. Following this idea you can easily set up a system to store all your keywords in a central database.

#### Step 3 :

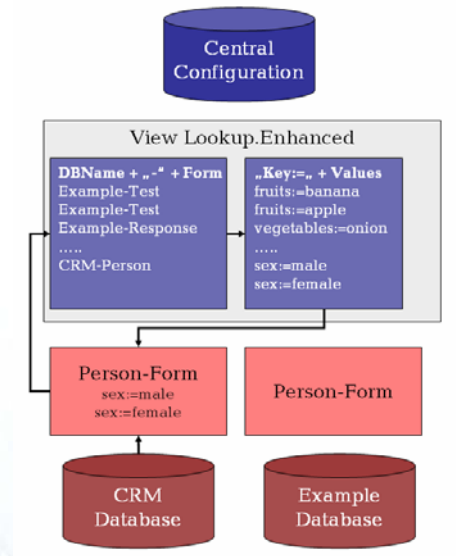
We create a new hidden multivalue field in the top of our form that is computed for display. This field will do a @DBLookup to our setup view and store all entries while the document is open for reading or editing – no matter if we really need these values.

```
REM {(c) Florian Lier - 2007_07_07};
REM {Here we can also take the value of another field};
_SRV:=@Subset(@DbName;1);
REM {Here we can easily point to a field containing the right path};
_PATH:=@Subset(@DbName;-1);
REM {Add your view-name here...};
_VIEW:="va_valuesenhanced";
REM {Here we can enter any key like a string or a field-content};
_KEY:=Form;
REM {In our case we have to point to the 2nd column - faster solution see final remarks};
_RETURN:=2;

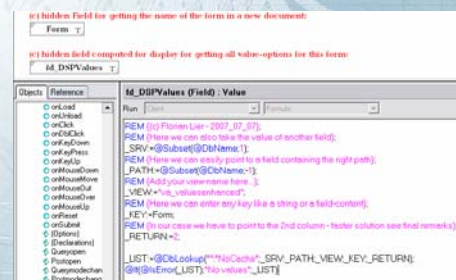
_LIST:=@DbLookup("":'NoCache";_SRV:_PATH:_VIEW;_KEY;_RETURN);
@if(@IsError(_LIST);"No values";_LIST)
```

#### Code for our setup field called « fd\_DSPValues »

If the result of your @DBLookup will exceed 64K you should split the content more granular and create a second field – but in general this should not happen at all.



Architecture for « global setup »



Field for storing the values (Step 3)



## Best Practices Lotus Domino - @Formulas I

### @DBLookup - @DBCColumn

Your fast and inconvenient access to data residing in Notes Databases



13.04.2008

#### Step 4

Instead of writing now a @DBLookup code we can use the following template for selecting values for our fields :

```
_KEY:=<FIELDNAME> OR "STRING";
_A:=fd_DSPValues;
_B:=@Right(_A;_KEY + ":=");
_C:=@Trim(@Unique(_B));
_D:=@If((@IsError(_C) | _KEY="");"No values found for your
selection";@Trim(@Unique(_C)));
```

\_D

Code for our setup field called « fd\_DSPValues »

#### Remarks

Getting the location of a database :

Using variables to tell your code where to point to (like database paths or server names) makes your code easier to read and debug. Adding a field to your form (computed for display) that is named by the « interface » (e. g. « fd\_PathCRM ») gives you a proper way of accessing location information and keep your code « slim ».

How to access data you are not allowed to read ?

Imagine you are setting up a workflow where each document needs to have a unique running number. And your users do not have access to other users requests. Then you can create a view where you will show the « Number » in the first categorized column.

Then you are able to perform a @DBCColumn to this view / column and retrieve the content of other documents – as long as the option « Don't show empty categories » is NOT selected.

Repeating @DBLookups / @DBCColumns

Pretty often dynamic tables are created with hide-when formulas (do not show the next row as long as this row is not used)... If you have included a field in each of these rows which is performing a @DBLookup, you can improve the speed of your application by stopping the execution of this code as long as some circumstances are not true :

```
REM {© Florian Lier – without the _EXIT row the @Dblookup will be submitted and will not
return a value as long as field « fd_ProjectAdd3 » is empty – now we save 4 seconds of time};
_KEY := fd_project + "." + fd_projectAdd3;
_EXIT:=@If(fd_ProjectAdd3="";@Return("");"");
```

```
_RESULT:=@DbLookup( "" ; _SERVER ; _DBPATH ; _VIEW ; _KEY ; _FIELD ) ;
```

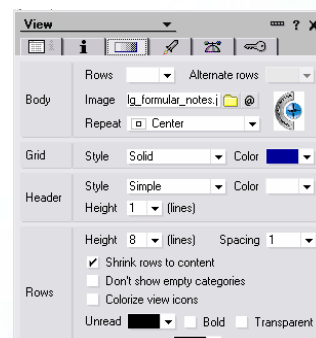
Code snippet for preventing unnecessary code being executed



Final version of new form (Step 4)



Final version of form with selection options (Step 4)



Accessing data you are not allowed to read (Remarks – 1)