

<h2 style="margin: 0;">Journal Entry</h2> <p>Created: 11.12 18:32 Modified: 11.12 18:43</p>	<p>Title: Reuse @Formula code with @ThisName, @Eval and @GetField</p> <p>Category: Notes &amp; Domino</p> <p>Diary date: 11.12.2006</p>
---	---

I've had it. I'm done with writing value list and default value formulas!

Since Notes 6, we have @ThisName. This formula, together with @Eval and @GetField open a whole new field of @Formula Programming: self configuring code. Code that adjusts to its current environment. That kind of code saves an awful lot of typing and testing, if you are careful to standardize the way you do things.

Let me propose to use:

- <FieldName> as the user interaction field. let's assume it is a "Dialog List".
  - <FieldName>\_values as the set of values (a list of some basic Notes data type)
  - <FieldName>\_default as the default value for the field on a new document
- As an added twist, we want to be able to specify the default value as either a literal value or a Notes @Formula. To determine if we have a formula or a literal, we enclose formulas with curly braces ("{" , "}").

For the default value formula in <FieldName>, use

```

_____
REM {Field independent formula -- assume default value is in <FieldName>_default.
Checks if a formula or a value is specified and evaluates formula if necessary};
_dflt := @GetField(@ThisName + "_default");
@if(@begins(_dflt;"{");
    @Eval(@Middle(_dflt ; "{" ; "}"));
    _dflt
)
_____

```

To have flexible configuration, I propose to use separate documents that define lists including list name (Field "Subject"), values ("ListValues"), default ("ListDefault"), type of configuration ("Type") and possibly other settings like "mandatory" or "validation formula".

Those documents are included in a configuration search view, "wwhConfigSearch". Our key is the Type (say, "ListConfig") concatenated with "\$\$\$" and the subject (say, "InternalAddress").

Now, we retrieve the value list with the following formula in <FieldName>\_values:

```
_____
    _subject := "InternalAddress";
    _view := "vwhConfigSearch";
    _type := "ListConfig";
    _field := "List" + @Right(@ThisName; "_");

    @DbLookup( "" : "Nocache";
              "" : "";
              _view;
              _type + "$$$" + _subject;
              _field;
              [FailSilent]
            )
_____
```

Now comes the first surprise: This very same formula can be reused unchanged to retrieve the default value in <FieldName>\_default!

Still, we have to edit the formula for another set of fields. So, won a lot, but not all. Therefore, let's up the ante a bit more: Let's say, our <FieldName> on the form is "InternalAddress", i.e. it is the same as the subject of the configuration document. Now, let's replace the \_subject line above by:

```
_____
    _subject := @Left(@ThisName; "_");
_____
```

Et voilà, as long as we have just one field using each of our list configurations, we can re-use the formula without changing a single character of it - the field names configure the formulas!

For me, this has been quite an eye opener. Naturally, there's more leeway: let's just make that subject field on the configuration document a text list and we have finally a way to configure things with a standardized set of formulas.

One word of caution: "Nocache" is a performance killer; use it wisely. It is good enough to have one field at the beginning of the form which specifies "Nocache" for its lookup, per configuration view. Once you have these views refreshed, there's no use to force that again -- thus, remove the other "Nocache" entries.

Have fun and program effectively!