|  |
| --- |
| Programberoende Rapporter |
| Dynamisk filtrering av LO/TO-poster med avseende på Rapportprogram |
|  |
| Orsak (Upprinnelse) till det hela och lösning med kodning i procedurer och delphikod. |
|  |

Programberoende Rapporter

Dynamisk filtrering av LO/TO-poster med avseende på Rapportprogram

# Kommentarer

• • •

# Upprinnelse

Lastorder(LO) och Transportorder rapporter filtreras med avseende på LO:ns resp TO:s men filtreringen önskas olika beroende från vilket program man kör rapporten från.

VIS\_ORDER vill se status:

* PRELIMINÄR
* NY
* ACCEPTERAD
* PROD KLAR

VISK vill se status:

* NY
* ACCEPTERAD
* PROD\_KLAR

Detta kan ej ske idag utan att man skapar nya procedurer och rapporter.

Detta kräver mer underhåll vilket ej är önskvärt.

Lösningen är att styra filtreringen i proceduren via en parameter som anger vilket program som kör rapporten.

# Proceduren

Ny parameter läggs till - @RUNNING\_PGM

ALTER PROCEDURE [dbo].[vis\_LO\_v3\_FR]

@ShippingPlanNo int,

@SupplierNo int = null

,@Language INT = 0 -- Swedish (1), English (other)

,@RUNNING\_PGM int = 0

Statusfiltreringen:

Vi testar bara mot att program VIS\_ORDER körs (@RUNNING\_PGM=1) då vi bara har 2 alternativ så länge. Kommer fler är det bara att utöka med fler tester.

Vet inte förnärvarande om status 9 och 10 skall vara med.. måste kollas.

WHERE CSH.ShippingPlanNo = @ShippingPlanNo

and SSP.ObjectType in (0,1,2)

AND ((SSP.SupplierNo = @SupplierNo) OR (@SupplierNo=-1))

AND((@RUNNING\_PGM=1

AND SSP.ShippingPlanStatus IN (0,1,3,6,9,10))

OR SSP.ShippingPlanStatus IN (1,3,6,9,10))

# FastReport koden

I Unit uReportController har följande kod ändrats med tillägg GRÖNT och borttagning RÖTT.

function TCMReportController.processParameters(aReportData: TCMMReportData;

aParams: TCMParams): TCMParams;

var

repParams: TCMParamsInfo;

newParams: TCMParams;

uCaseParams: TCMParams;

pName: string;

i: integer;

begin

// Create containers for intermediate processing

newParams := TCMParams.create(); // Where our result will be

uCaseParams := TCMParams.create(); // Uppercase version of input parameters

repParams := aReportData.getAllParameters;

// Parameter info from used stored procedures

// Populate the uppercase container

for pName in aParams.keys do

uCaseParams.Add(AnsiUpperCase(pName), aParams.Items[pName]);

// Check if the uppercase name from input parameters

// can be found as an uppercase version from the stord procedures

// and add the input parameter with the name of the SP-version

// if it was found.

for pName in repParams.keys do

begin

If uCaseParams.ContainsKey(AnsiUpperCase(pName)) then

begin

newParams.Add(pName, uCaseParams.Items[AnsiUpperCase(pName)]);

end

else if pName = '@RUNNING\_PGM' then

newParams.Add(pName, FRConstants.RunningPgm)

else

begin

(\* ShowMessage('Parameter "' + pName + '" not found!');

Result := nil;

exit;

\*)

end;

end;

Result := newParams;

end;

procedure TCMReportController.addParams(var aSP: TFDStoredProc;

aParams: TCMParams);

var

wantedParams: TCMParamsInfo;

i: integer;

key: string;

Value: variant;

begin

wantedParams := getParamsInfo(aSP.StoredProcName);

for key in wantedParams.keys do

begin

if aParams.ContainsKey(key) then

begin

aParams.TryGetValue(key, Value);

aSP.Params.ParamByName(key).Value := Value;

end

else if key = '@RUNNING\_PGM' then

aSP.Params.ParamByName(key).Value := FRConstants.RunningPgm;

end;

end;