DESIGN PATTERNS IN NAV 2016

Nikola Kukrika, Ciprian Iordache, Gary Winter
(Microsoft MDCC / Cloud Ready Software)

1. Events

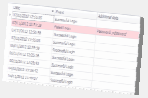
2. Passwords and sensitive data

3. Variant façade

4. Data-driven blocked entity

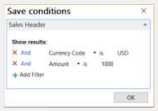
5. Try method

6. Logging

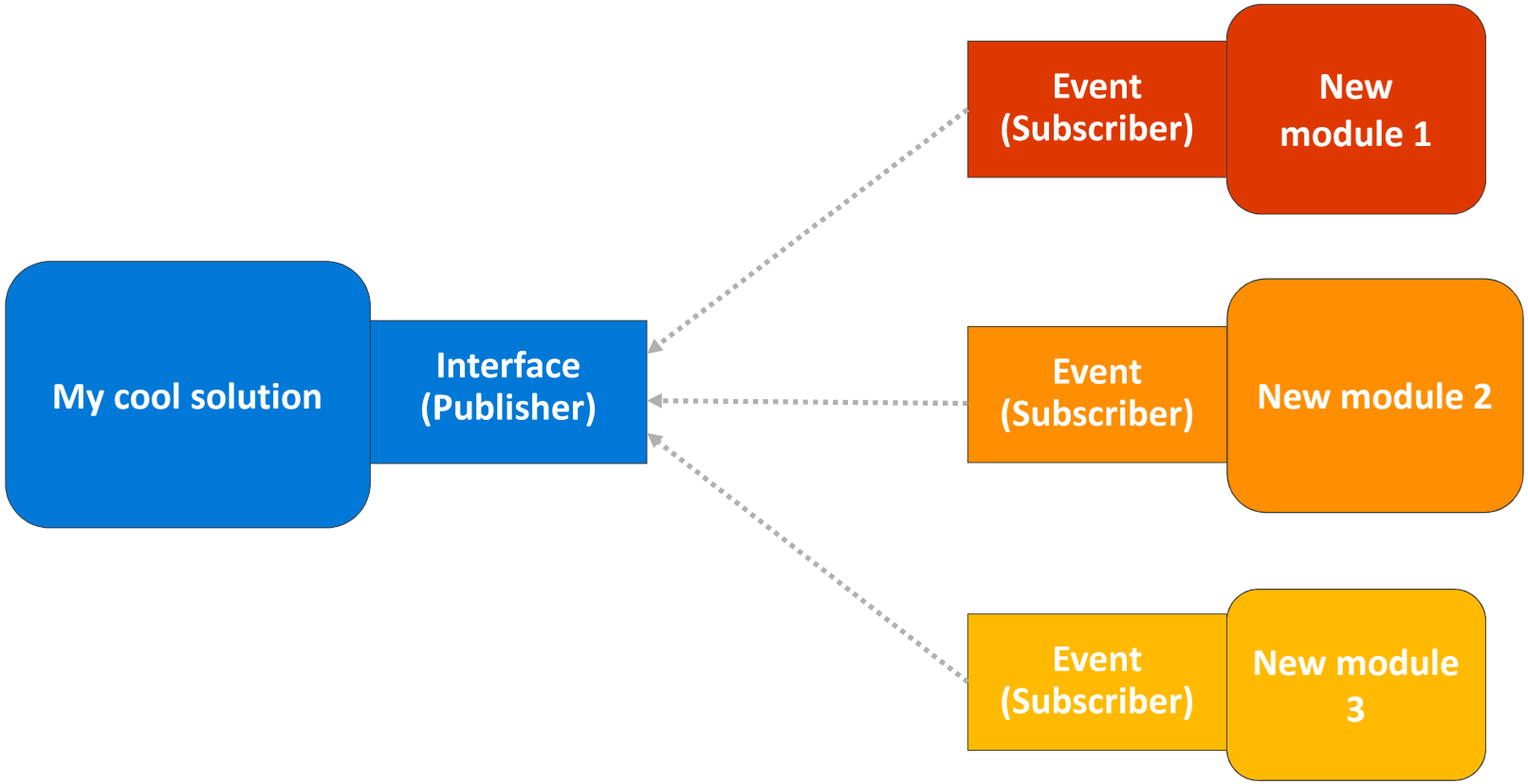


7. Error message processing

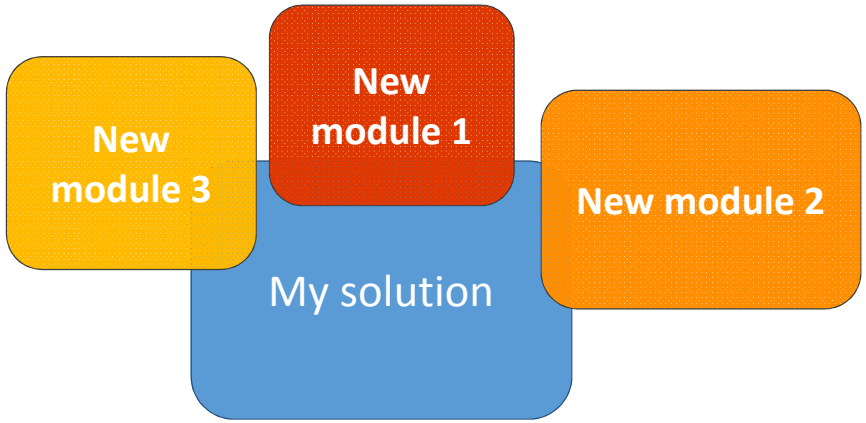
8. Dynamic request pages



Event and Interface



Event and Interface



Problem statement

~~I have to modify my solution whenever a new module gets added.~~

Solution

Use event and interface

Event and Interface

For service connections we have applied the event and interface pattern

Service Connections

Description	Host Name	Status
Bank Data Conv. Service Set...	https://nav.amcbanking.com/na...	Enabled
CRM Connection Setup		Disabled
Yahoo Currency Exchange R...	http://query.yahooapis.com/v1/p...	Disabled
Doc. Exch. Service Setup	https://api-sandbox.tradeshift.co...	Enabled
OCR Service Setup		Disabled
Online Map Setup		Enabled
SMTP Mail Setup		Disabled
Social Engagement Setup		Disabled

Interface (Publisher)

```
OnOpenPage()
OnRegisterServiceConnection(Rec);

[IntegrationEvent] OnRegisterServiceConnection(VAR ServiceConnection : Record "Service Connection")
```

Property	Value
ID	**1**
Local	<No>
TryFunction	<No>
Events	**Publisher**
EventType	**Integration**
IncludeSender	<No>
GlobalVarAccess	<No>

Event Subscriber

```
[EventSubscriber] HandleOCRRegisterServiceConnection(VAR ServiceConnection : Record "Service Connection")
OCRServiceSetup.GET;
RecRef.GETTABLE(OCRServiceSetup);

IF OCRServiceSetup.Enabled THEN
  ServiceConnection.Status := ServiceConnection.Status::Enabled
ELSE
  ServiceConnection.Status := ServiceConnection.Status::Disabled;
WITH OCRServiceSetup DO
  ServiceConnection.InsertServiceConnection(
    ServiceConnection,RecRef.RECORDID,TABLENAME,"Service URL",PAGE::"OCR Service Setup");
```

Property	Value
ID	**20**
Local	<No>
TryFunction	<No>
Event	**Subscriber**
EventPublisherObject	**Table Service Connection**
EventFunction	**OnRegisterServiceConnection**
OnMissingLicense	<Error>
OnMissingPermission	<Error>

Event and Interface

Cons

Write protective code

Pros

Loose coupling with clear responsibility

Pluggable design

Passwords and sensitive data

Passwords and sensitive data

Problem statement

1. Credit cards stored randomly
2. Everyone has permission to access
3. Not pin protected

The electronic “wallet”

Passwords and sensitive data

Problem statement

1. ~~Credit cards stored randomly~~
2. ~~Everyone has permission to access~~
3. ~~Not pin protected~~

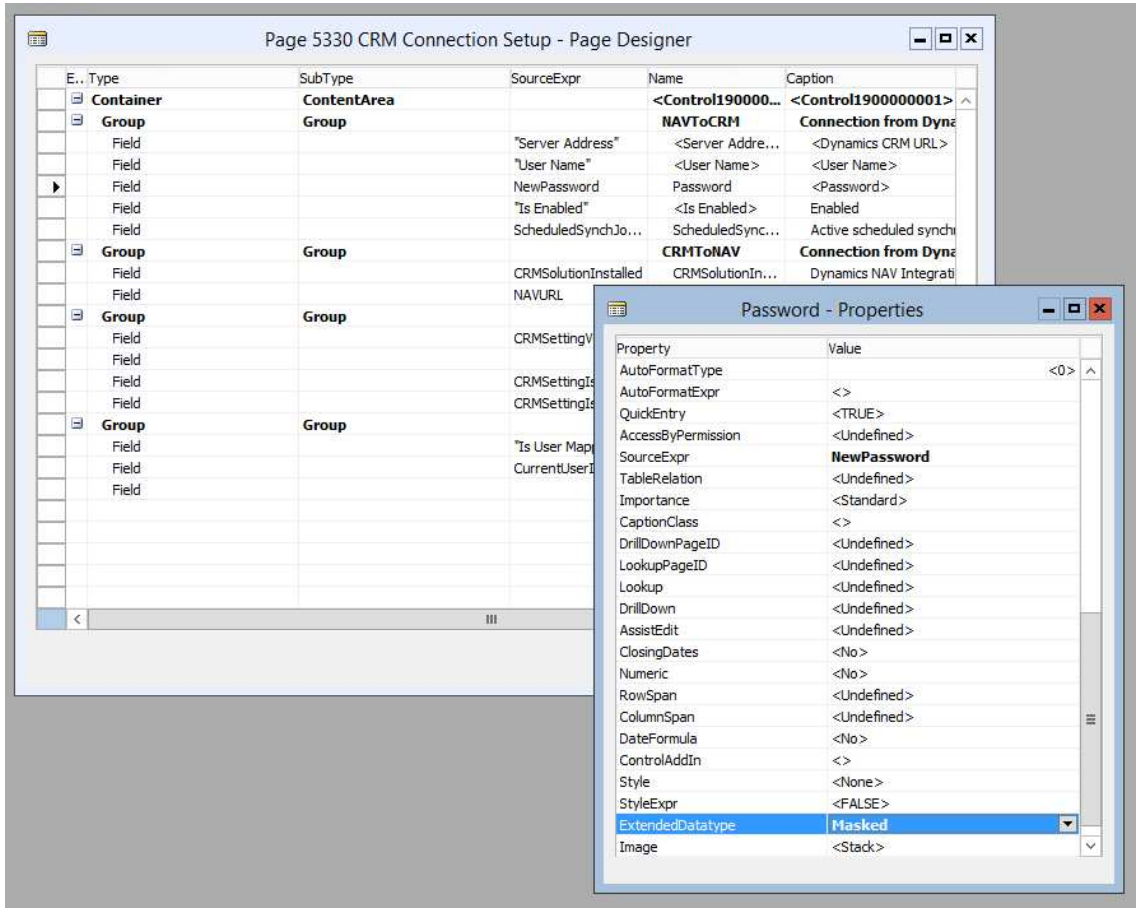
Solution

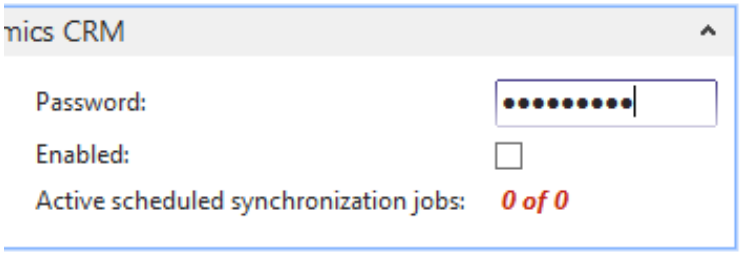
1. Encapsulate sensitive data
2. Provide a single point of access
3. Encrypt data

Demo

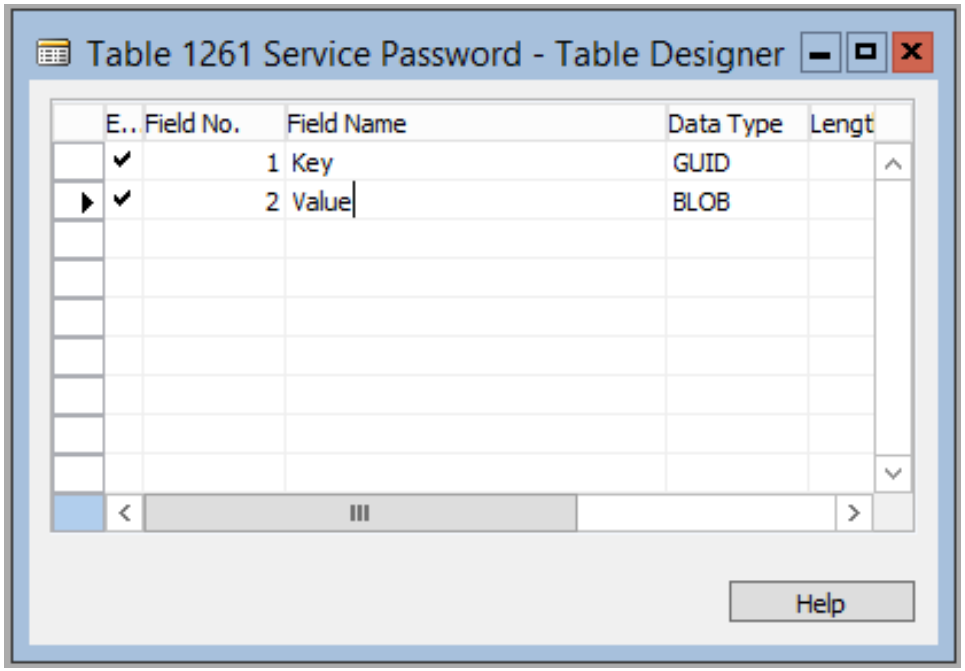
Passwords and sensitive data

“Mask” sensitive data in UI

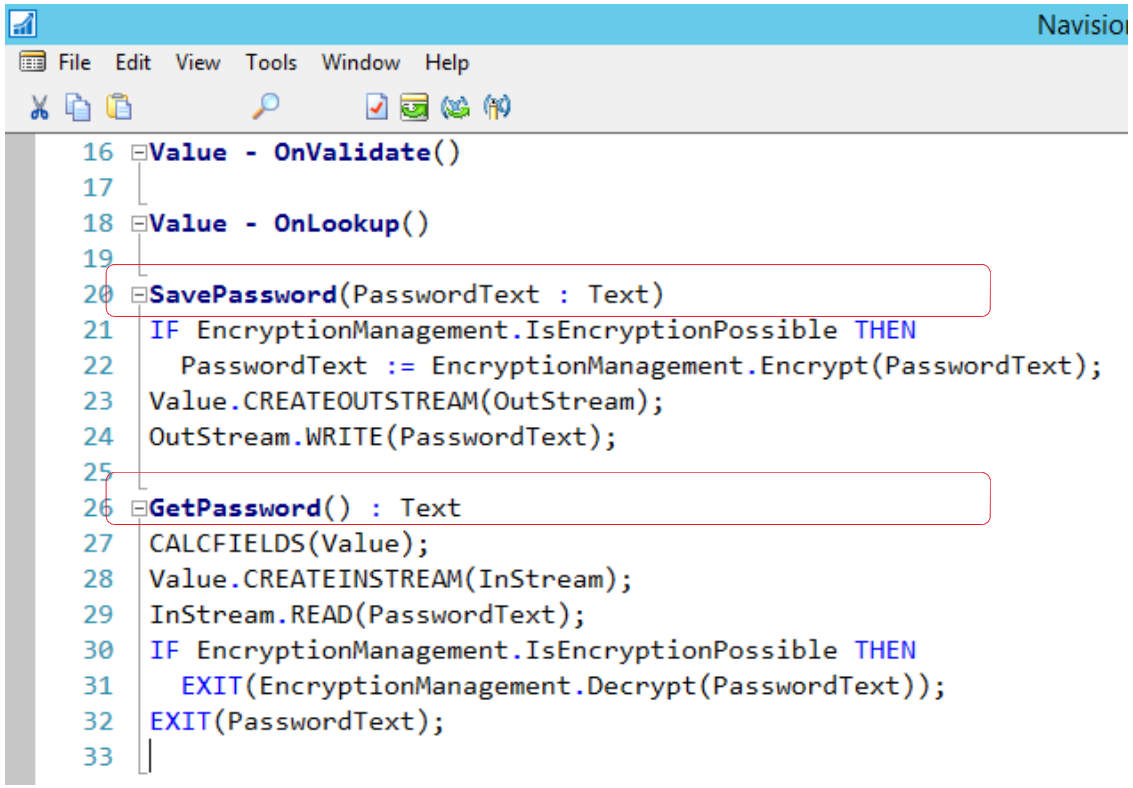




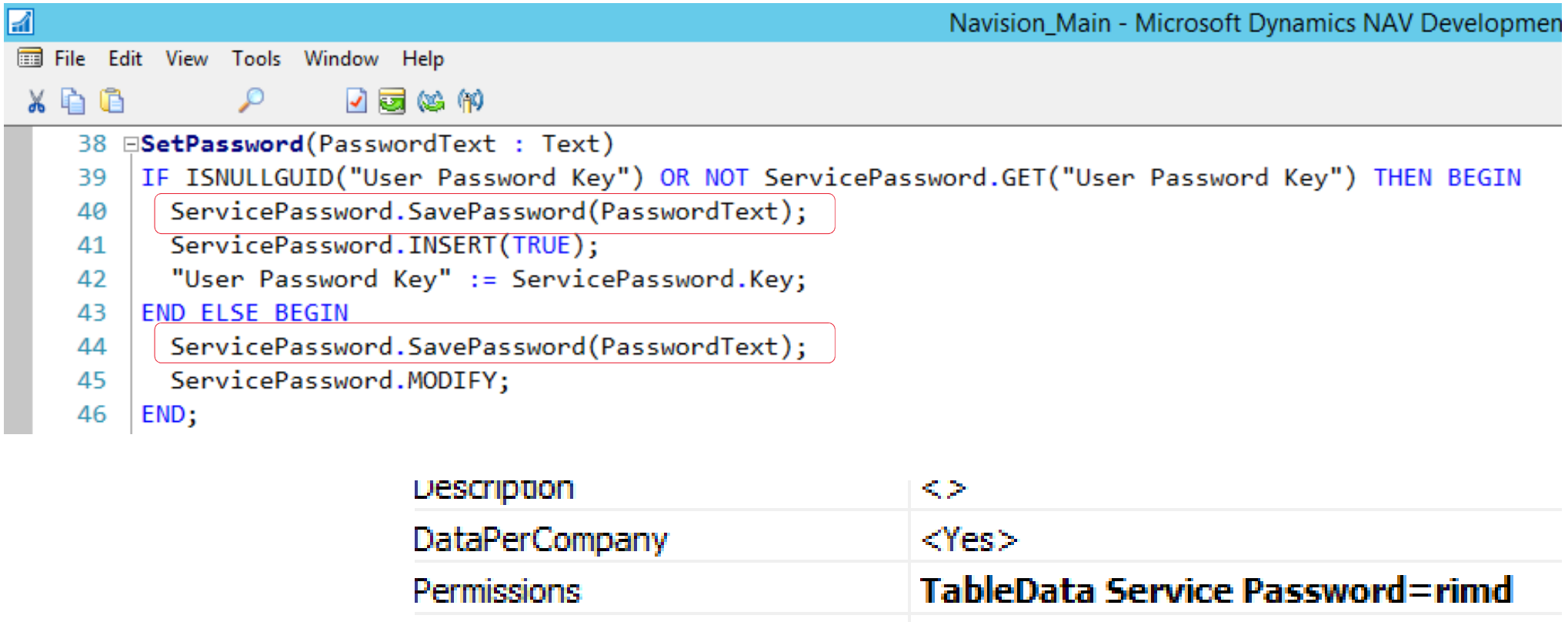
Encapsulate sensitive data



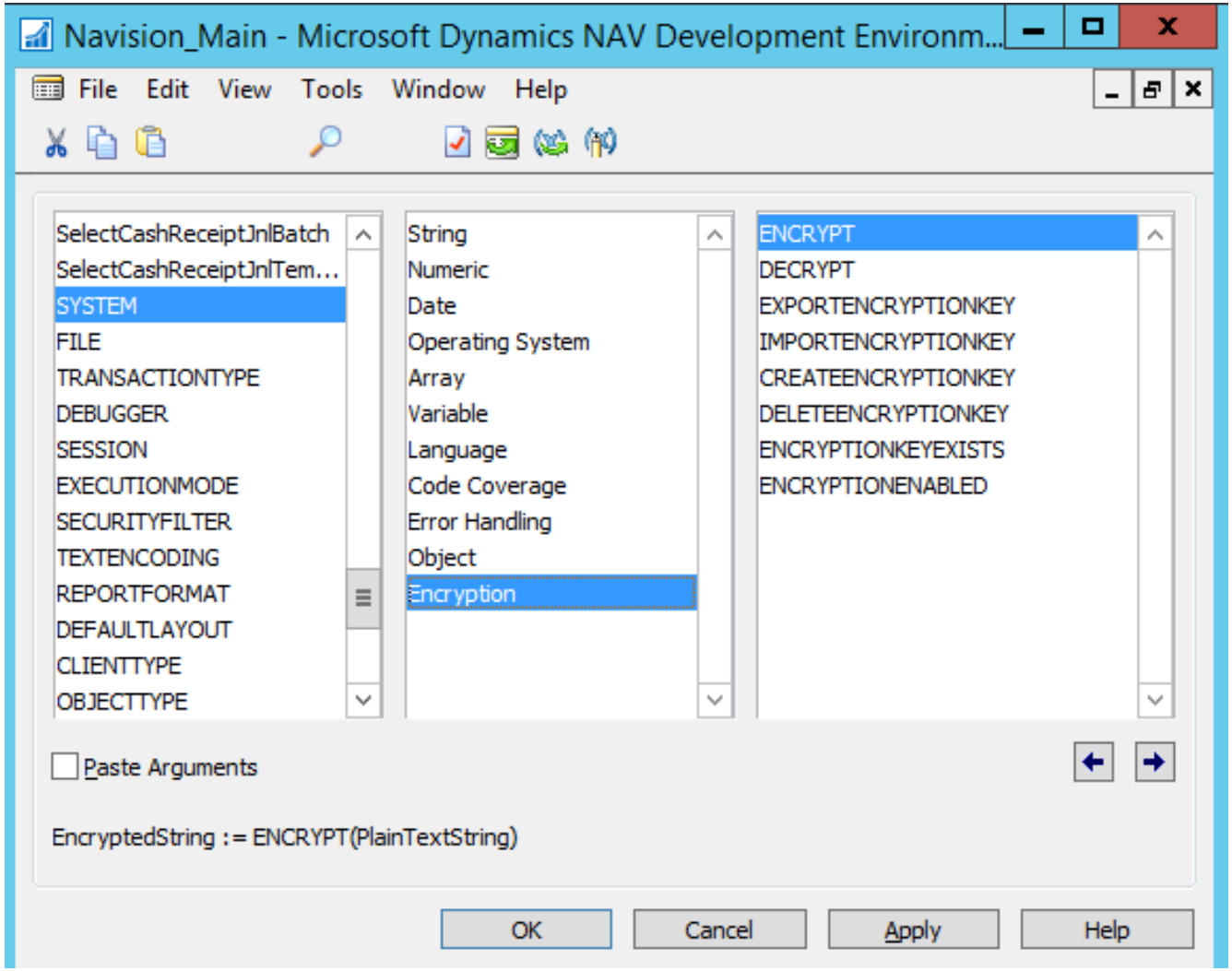
Single point of access



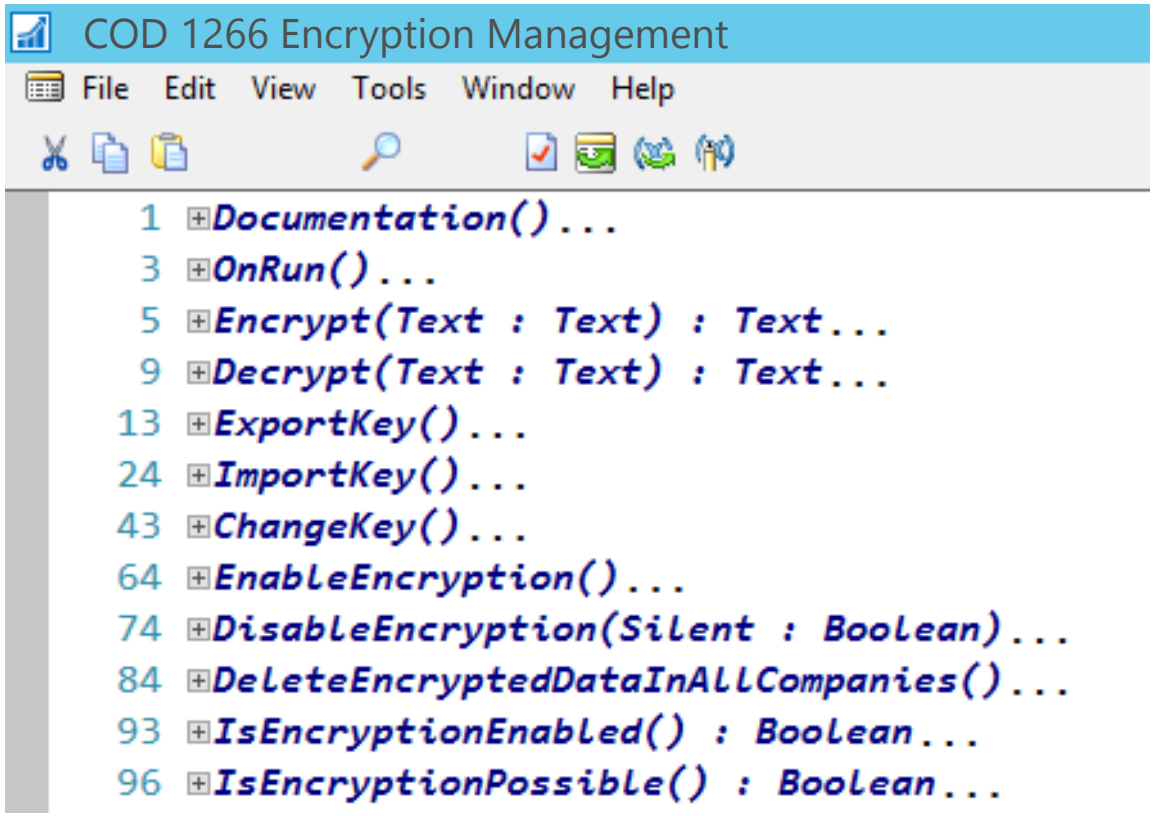
Single point of access



Encryption


Navision_Main - Microsoft Dynamics NAV Development Environm...
File Edit View Tools Window Help
SelectCashReceiptJnlBatch
SelectCashReceiptJnlTem...
SYSTEM
FILE
TRANSACTIONTYPE
DEBUGGER
SESSION
EXECUTIONMODE
SECURITYFILTER
TEXTENCODING
REPORTFORMAT
DEFAULTLAYOUT
CLIENTTYPE
OBJECTTYPE
String
Numeric
Date
Operating System
Array
Variable
Language
Code Coverage
Error Handling
Object
Encryption
ENCRYPT
DECRYPT
EXPORTENCRYPTIONKEY
IMPORTENCRYPTIONKEY
CREATEENCRYPTIONKEY
DELETEENCRYPTIONKEY
ENCRYPTIONKEYEXISTS
ENCRYPTIONENABLED
Paste Arguments
EncryptedString := ENCRYPT(PlainTextString)
OK
Cancel
Apply
Help

Encryption library



Encourage the user to enable encryption

```
LOCAL PROCEDURE CheckEncryption@6();
   BEGIN
     IF NOT ENCRYPTIONENABLED THEN
       IF CONFIRM(EncryptionIsNotActivatedQst) THEN
         PAGE.RUN(PAGE::"Data Encryption Management");
   END;
```

Summary

- Encapsulate sensitive data
- Protect access through proxy library
- Encrypt data

Passwords and Sensitive Data Pattern

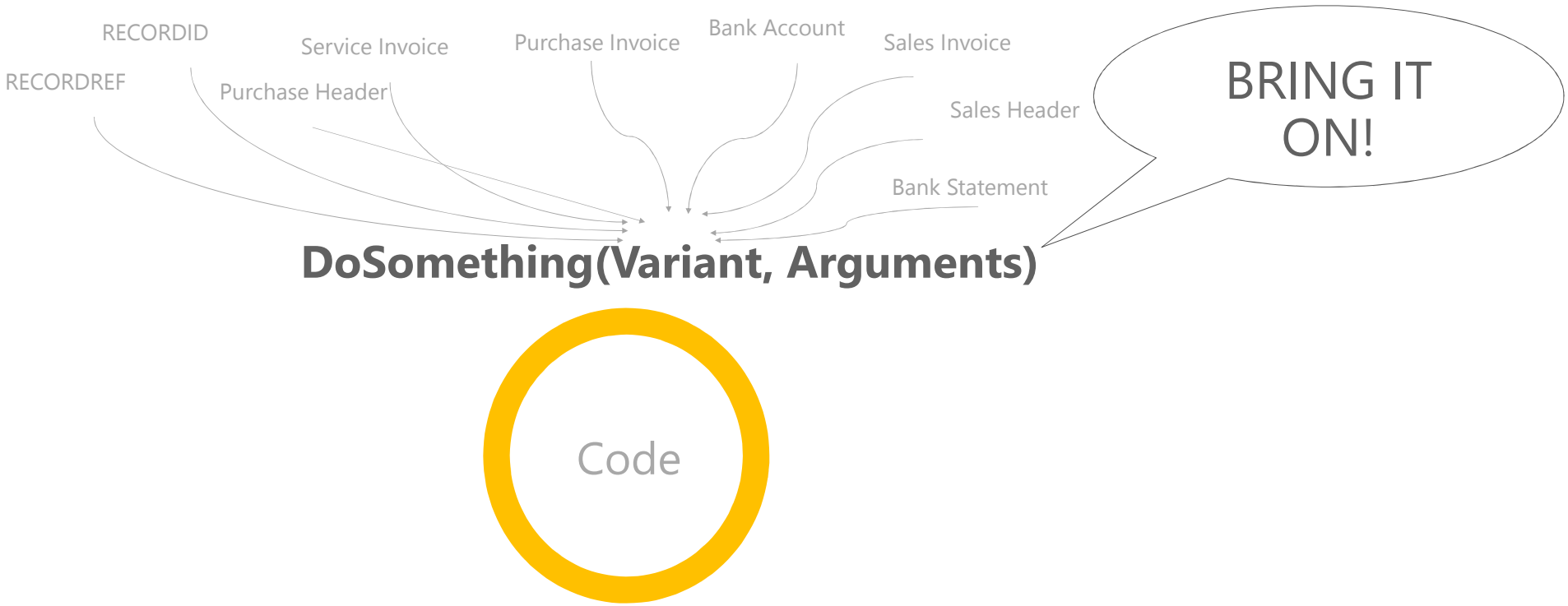
The Good

- Highly reusable and scalable
- Keeps the sensitive data in a secure way

The Bad

- Texts of max 250 chars can be encrypted

Variant façade



Variant façade

Problem

The existing code must be extended to support a new record

=> copy old code, change a few aspects

=> often more than 90% code is duplicated

Variant façade

Spot the difference

```
PrintPurchHeader(PurchHeader : Record "Purchase Header")
PurchHeader.SETRANGE("No.",PurchHeader."No.");
PurchSetup.GET;
IF PurchSetup."Calc. Inv. Discount" THEN BEGIN
  PurchLine.RESET;
  PurchLine.SETRANGE("Document Type",PurchHeader."Document Type");
  PurchLine.SETRANGE("Document No.",PurchHeader."No.");
  PurchLine.FINDFIRST;
  PurchCalcDisc.RUN(PurchLine);
  PurchHeader.GET(PurchHeader."Document Type",PurchHeader."No.");
  COMMIT;
END;
CASE PurchHeader."Document Type" OF
  PurchHeader."Document Type"::Quote:
    ReportSelection.SETRANGE(Usage,ReportSelection.Usage::"P.Quote");
  PurchHeader."Document Type"::"Blanket Order":
    ReportSelection.SETRANGE(Usage,ReportSelection.Usage::"P.Blanket")
  PurchHeader."Document Type"::Order:
    ReportSelection.SETRANGE(Usage,ReportSelection.Usage::"P.Order");
  PurchHeader."Document Type"::"Return Order":
    ReportSelection.SETRANGE(Usage,ReportSelection.Usage::"P.Return");
  ELSE
    EXIT;
END;
ReportSelection.SETFILTER("Report ID",'<>0');
ReportSelection.FIND('-');
REPEAT
  REPORT.RUNMODAL(ReportSelection."Report ID",TRUE,FALSE,PurchHeader)
UNTIL ReportSelection.NEXT = 0;
```

```
PrintServiceHeader(ServiceHeader : Record "Service Header")
SalesSetup.GET;
ServiceHeader.SETRANGE("No.",ServiceHeader."No.");
IF SalesSetup."Calc. Inv. Discount" THEN BEGIN
  ServLine.RESET;
  ServLine.SETRANGE("Document Type",ServiceHeader."Document Type");
  ServLine.SETRANGE("Document No.",ServiceHeader."No.");
  ServLine.FINDFIRST;
  ServCalcDisc.RUN(ServLine);
  ServiceHeader.GET(ServiceHeader."Document Type",ServiceHeader."No.");
  COMMIT;
END;
CASE ServiceHeader."Document Type" OF
  ServiceHeader."Document Type"::Quote:
    ReportSelection.SETRANGE(Usage,ReportSelection.Usage::"SM.Quote");
  ServiceHeader."Document Type"::Order:
    ReportSelection.SETRANGE(Usage,ReportSelection.Usage::"SM.Order");
  ServiceHeader."Document Type"::Invoice:
    ReportSelection.SETRANGE(Usage,ReportSelection.Usage::"SM.Invoice");
  ServiceHeader."Document Type"::"Credit Memo":
    ReportSelection.SETRANGE(Usage,ReportSelection.Usage::"SM.Credit Memo");
  ELSE
    EXIT;
END;
ReportSelection.SETFILTER("Report ID",'<>0');
IF ReportSelection.FIND('-') THEN BEGIN
  REPEAT
    REPORT.RUNMODAL(ReportSelection."Report ID",TRUE,FALSE,ServiceHeader);
  UNTIL ReportSelection.NEXT = 0;
END ELSE
  ERROR(Text002,ReportSelection.FIELDCAPTION("Report ID"),
    ServiceHeader.TABLECAPTION,ReportSelection.TABLECAPTION);
```

Variant façade

Spot the difference – Master level

Document Print - 12 Functions

```
PrintPurchHeader(PurchHeader : Record "Purchase Header")
PurchHeader.SETRANGE("No.",PurchHeader."No.");
PurchSetup.GET;
IF PurchSetup."Calc. Inv. Discount" THEN BEGIN
  PurchLine.RESET;
  PurchLine.SETRANGE("Document Type",PurchHeader."Document Type");
  PurchLine.SETRANGE("Document No.",PurchHeader."No.");
  PurchLine.FINDFIRST;
  PurchCalcDisc.RUN(PurchLine);
  PurchHeader.GET(PurchHeader."Document Type",PurchHeader."No.");
  COMMIT;
END;
CASE PurchHeader."Document Type" OF
  PurchHeader."Document Type"::Quote:
    ReportSelection.SETRANGE(Usage,ReportSelection.Usage::"P.Quote");
  PurchHeader."Document Type"::"Blanket Order":
    ReportSelection.SETRANGE(Usage,ReportSelection.Usage::"P.Blanket")
  PurchHeader."Document Type"::Order:
    ReportSelection.SETRANGE(Usage,ReportSelection.Usage::"P.Order");
  PurchHeader."Document Type"::"Return Order":
    ReportSelection.SETRANGE(Usage,ReportSelection.Usage::"P.Return");
  ELSE
    EXIT;
END;
ReportSelection.SETFILTER("Report ID",'<>0');
ReportSelection.FIND('-');
REPEAT
  REPORT.RUNMODAL(ReportSelection."Report ID",TRUE,FALSE,PurchHeader)
UNTIL ReportSelection.NEXT = 0;
```

```
PrintBankAccStmt(BankAccStmt : Record "Bank Account Statement")
BankAccStmt.SETRECFILTER;
ReportSelection.SETRANGE(Usage,ReportSelection.Usage::"B.Stmt");
ReportSelection.SETFILTER("Report ID",'<>0');
ReportSelection.ASCENDING := FALSE;
ReportSelection.FIND('-');
REPEAT
  REPORT.RUN(ReportSelection."Report ID",TRUE,FALSE,BankAccStmt);
UNTIL ReportSelection.NEXT = 0;

PrintCheck(VAR NewGenJnlLine : Record "Gen. Journal Line")
GenJnlLine.COPY(NewGenJnlLine);
GenJnlLine.OnCheckGenJournalLinePrintCheckRestrictions;
ReportSelection.SETRANGE(Usage,ReportSelection.Usage::"B.Check");
ReportSelection.SETFILTER("Report ID",'<>0');
ReportSelection.FIND('-');
REPEAT
  REPORT.RUNMODAL(ReportSelection."Report ID",TRUE,FALSE,GenJnlLine);
UNTIL ReportSelection.NEXT = 0;
```

```
PrintSalesHeaderArch(SalesHeaderArch : Record "Sales Header Archive")
SalesHeaderArch.SETRANGE("No.",SalesHeaderArch."No.");
SalesHeaderArch.SETRANGE("Doc. No. Occurrence",SalesHeaderArch."Doc. No. Occurrence");
SalesHeaderArch.SETRANGE("Version No.",SalesHeaderArch."Version No.");
CASE SalesHeaderArch."Document Type" OF
  SalesHeaderArch."Document Type"::Quote:
    BEGIN
      ReportSelection.SETRANGE(Usage,ReportSelection.Usage::"S.Arch. Quote");
      SalesHeaderArch.SETRANGE("Document Type",SalesHeaderArch."Document Type"::Quote);
    END;
  SalesHeaderArch."Document Type"::Order:
    BEGIN
      ReportSelection.SETRANGE(Usage,ReportSelection.Usage::"S.Arch. Order");
      SalesHeaderArch.SETRANGE("Document Type",SalesHeaderArch."Document Type"::Order);
    END;
  SalesHeaderArch."Document Type"::"Return Order":
    BEGIN
      ReportSelection.SETRANGE(Usage,ReportSelection.Usage::"S. Arch. Return Order");
      SalesHeaderArch.SETRANGE("Document Type",SalesHeaderArch."Document Type"::"Return Order");
    END;
  ELSE
    EXIT;
END;
ReportSelection.SETFILTER("Report ID",'<>0');
ReportSelection.FIND('-');
REPEAT
  REPORT.RUNMODAL(ReportSelection."Report ID",TRUE,FALSE,SalesHeaderArch)
UNTIL ReportSelection.NEXT = 0;
```

```
PrintPurchHeaderArch(PurchHeaderArch : Record "Purchase Header Archive")
PurchHeaderArch.SETRANGE("No.",PurchHeaderArch."No.");
PurchHeaderArch.SETRANGE("Doc. No. Occurrence",PurchHeaderArch."Doc. No. Occurrence");
PurchHeaderArch.SETRANGE("Version No.",PurchHeaderArch."Version No.");

CASE PurchHeaderArch."Document Type" OF
  PurchHeaderArch."Document Type"::Quote:
    BEGIN
      ReportSelection.SETRANGE(Usage,ReportSelection.Usage::"P.Arch. Quote");
      PurchHeaderArch.SETRANGE("Document Type",PurchHeaderArch."Document Type"::Quote);
    END;
  PurchHeaderArch."Document Type"::Order:
    BEGIN
      ReportSelection.SETRANGE(Usage,ReportSelection.Usage::"P.Arch. Order");
      PurchHeaderArch.SETRANGE("Document Type",PurchHeaderArch."Document Type"::Order);
    END;
  PurchHeaderArch."Document Type"::"Return Order":
    BEGIN
      ReportSelection.SETRANGE(Usage,ReportSelection.Usage::"P. Arch. Return Order");
      PurchHeaderArch.SETRANGE("Document Type",PurchHeaderArch."Document Type"::"Return Order");
    END;
  ELSE
    EXIT;
END;
ReportSelection.SETFILTER("Report ID",'<>0');
ReportSelection.FIND('-');
REPEAT
  REPORT.RUNMODAL(ReportSelection."Report ID",TRUE,FALSE,PurchHeaderArch)
UNTIL ReportSelection.NEXT = 0;
```

```
DoPrintSalesHeader(SalesHeader : Record "Sales Header";SendAsEmail : Boolean)
SalesHeader.SETRANGE("No.",SalesHeader."No.");
SalesSetup.GET;
IF SalesSetup."Calc. Inv. Discount" THEN BEGIN
  SalesLine.RESET;
  SalesLine.SETRANGE("Document Type",SalesHeader."Document Type");
  SalesLine.SETRANGE("Document No.",SalesHeader."No.");
  SalesLine.FINDFIRST;
  SalesCalcDisc.RUN(SalesLine);
  SalesHeader.GET(SalesHeader."Document Type",SalesHeader."No.");
  COMMIT;
END;
CASE SalesHeader."Document Type" OF
  SalesHeader."Document Type"::Quote:
    ReportSelections.SETRANGE(Usage,ReportSelections.Usage::"S.Quote");
  SalesHeader."Document Type"::"Blanket Order":
    ReportSelections.SETRANGE(Usage,ReportSelections.Usage::"S.Blanket");
  SalesHeader."Document Type"::Order:
    ReportSelections.SETRANGE(Usage,ReportSelections.Usage::"S.Order");
  SalesHeader."Document Type"::"Return Order":
    ReportSelections.SETRANGE(Usage,ReportSelections.Usage::"S.Return");
  SalesHeader."Document Type"::Invoice:
    ReportSelections.SETRANGE(Usage,ReportSelections.Usage::"S.Invoice");
  SalesHeader."Document Type"::"Credit Memo":
    ReportSelections.SETRANGE(Usage,ReportSelections.Usage::"S.Cr.Memo");
  ELSE
    EXIT;
END;

IF CustomReportSelection.PrintCustomReports(SalesHeader,SendAsEmail,FALSE) = 0 THEN BEGIN
  ReportSelections.SETFILTER("Report ID",'<>0');
  ReportSelections.FIND('-');
  REPEAT
    IF SendAsEmail THEN BEGIN
      AttachmentFilePath := SaveSalesHeaderReportAsPdf(SalesHeader,ReportSelections."Report ID");
      DocumentMailing.EmailFileFromSalesHeader(SalesHeader,AttachmentFilePath,CustomReportSelection);
    END ELSE
      REPORT.RUNMODAL(ReportSelections."Report ID",TRUE,FALSE,SalesHeader)
  UNTIL ReportSelections.NEXT = 0;
END;
```

```
PrintAsmHeader(AsmHeader : Record "Assembly Header")
AsmHeader.SETRANGE("No.",AsmHeader."No.");
CASE AsmHeader."Document Type" OF
  AsmHeader."Document Type"::Quote,
  AsmHeader."Document Type"::"Blanket Order",
  AsmHeader."Document Type"::Order:
    ReportSelections.SETRANGE(Usage,ReportSelections.Usage::"Asm. Order");
  ELSE
    EXIT;
END;
ReportSelections.SETFILTER("Report ID",'<>0');
ReportSelections.FIND('-');
REPEAT
  REPORT.RUNMODAL(ReportSelections."Report ID",TRUE,FALSE,AsmHeader)
UNTIL ReportSelections.NEXT = 0;

PrintSalesOrder(SalesHeader : Record "Sales Header";Usage : 'Order Confirmation,Work Order,Pick Instruction')
IF SalesHeader."Document Type" <> SalesHeader."Document Type"::Order THEN
  EXIT;

SalesHeader.SETRANGE("No.",SalesHeader."No.");
SalesSetup.GET;
IF SalesSetup."Calc. Inv. Discount" THEN BEGIN
  SalesLine.RESET;
  SalesLine.SETRANGE("Document Type",SalesHeader."Document Type");
  SalesLine.SETRANGE("Document No.",SalesHeader."No.");
  SalesLine.FINDFIRST;
  SalesCalcDisc.RUN(SalesLine);
  SalesHeader.GET(SalesHeader."Document Type",SalesHeader."No.");
  COMMIT;
END;

CASE Usage OF
  Usage::"Order Confirmation":
    ReportSelection.SETRANGE(Usage,ReportSelection.Usage::"S.Order");
  Usage::"Work Order":
    ReportSelection.SETRANGE(Usage,ReportSelection.Usage::"S.Work Order");
  Usage::"Pick Instruction":
    ReportSelection.SETRANGE(Usage,ReportSelection.Usage::"S.Order Pick Instruction");
  ELSE
    EXIT;
END;

IF CustomReportSelection.PrintCustomReports(SalesHeader,FALSE,FALSE) = 0 THEN BEGIN
  ReportSelection.SETFILTER("Report ID",'<>0');
  ReportSelection.FIND('-');
  REPEAT
    REPORT.RUNMODAL(ReportSelection."Report ID",GUIALLOWED,FALSE,SalesHeader)
  UNTIL ReportSelection.NEXT = 0;
END;
```

```
PrintServiceHeader(ServiceHeader : Record "Service Header")
SalesSetup.GET;
ServiceHeader.SETRANGE("No.",ServiceHeader."No.");
IF SalesSetup."Calc. Inv. Discount" THEN BEGIN
  ServLine.RESET;
  ServLine.SETRANGE("Document Type",ServiceHeader."Document Type");
  ServLine.SETRANGE("Document No.",ServiceHeader."No.");
  ServLine.FINDFIRST;
  ServCalcDisc.RUN(ServLine);
  ServiceHeader.GET(ServiceHeader."Document Type",ServiceHeader."No.");
  COMMIT;
END;
CASE ServiceHeader."Document Type" OF
  ServiceHeader."Document Type"::Quote:
    ReportSelection.SETRANGE(Usage,ReportSelection.Usage::"SM.Quote");
  ServiceHeader."Document Type"::Order:
    ReportSelection.SETRANGE(Usage,ReportSelection.Usage::"SM.Order");
  ServiceHeader."Document Type"::Invoice:
    ReportSelection.SETRANGE(Usage,ReportSelection.Usage::"SM.Invoice");
  ServiceHeader."Document Type"::"Credit Memo":
    ReportSelection.SETRANGE(Usage,ReportSelection.Usage::"SM.Credit Memo");
  ELSE
    EXIT;
END;
ReportSelection.SETFILTER("Report ID",'<>0');
IF ReportSelection.FIND('-') THEN BEGIN
  REPEAT
    REPORT.RUNMODAL(ReportSelection."Report ID",TRUE,FALSE,ServiceHeader);
  UNTIL ReportSelection.NEXT = 0;
END ELSE
  ERROR(Text002,ReportSelection.FIELDCAPTION("Report ID"),
    ServiceHeader.TABLECAPTION,ReportSelection.TABLECAPTION);

PrintTransferHeader(TransHeader : Record "Transfer Header")
TransHeader.SETRANGE("No.",TransHeader."No.");
ReportSelection.SETRANGE(Usage,ReportSelection.Usage::Inv1);
ReportSelection.SETFILTER("Report ID",'<>0');
ReportSelection.FIND('-');
REPEAT
  REPORT.RUNMODAL(ReportSelection."Report ID",TRUE,FALSE,TransHeader)
UNTIL ReportSelection.NEXT = 0;
```

```
PrintPurchHeader(PurchHeader : Record "Purchase Header")
PurchHeader.SETRANGE("No.",PurchHeader."No.");
PurchSetup.GET;
IF PurchSetup."Calc. Inv. Discount" THEN BEGIN
  PurchLine.RESET;
  PurchLine.SETRANGE("Document Type",PurchHeader."Document Type"),
  PurchLine.SETRANGE("Document No.",PurchHeader."No.");
  PurchLine.FINDFIRST;
  PurchCalcDisc.RUN(PurchLine);
  PurchHeader.GET(PurchHeader."Document Type",PurchHeader."No.");
  COMMIT;
END;
CASE PurchHeader."Document Type" OF
  PurchHeader."Document Type"::Quote:
    ReportSelection.SETRANGE(Usage,ReportSelection.Usage::"P.Quote");
  PurchHeader."Document Type"::"Blanket Order":
    ReportSelection.SETRANGE(Usage,ReportSelection.Usage::"P.Blanket");
  PurchHeader."Document Type"::Order:
    ReportSelection.SETRANGE(Usage,ReportSelection.Usage::"P.Order");
  PurchHeader."Document Type"::"Return Order":
    ReportSelection.SETRANGE(Usage,ReportSelection.Usage::"P.Return");
  ELSE
    EXIT;
END;
ReportSelection.SETFILTER("Report ID",'<>0');
ReportSelection.FIND('-');
REPEAT
  REPORT.RUNMODAL(ReportSelection."Report ID",TRUE,FALSE,PurchHeader)
UNTIL ReportSelection.NEXT = 0;
```

Variant façade

Spot the difference – Jedi level

20 Records / Functions

Variant façade

Problem

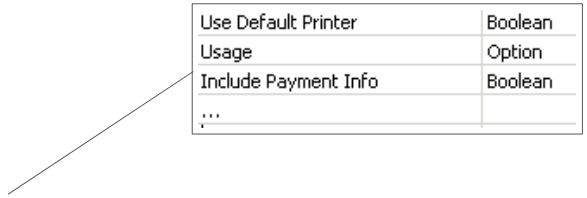
- To support a new record often more than 90% code is duplicated
- With each record added code is becoming worse:
- Multiple flows
- Harder to understand
- Harder to maintain
- Harder to extend / upgrade
- Harder to Test
- Cost of adding the new table type is always the same

Solution

- Provide a single interface for all of the record types
- Make a Clear Separation between Table Specific Code from Common code
- Adding a new record should not require code changes or they should be minimal

Variant façade

Signature




```
PrintDocument(RecordVariant : Variant;PrintDocumentArguments : Record "Print Document Arguments")
```

Variant façade

Use the variant

```
PrintDocument(RecordVariant : Variant;PrintDocumentArguments : Record "Print Document Arguments")
REPORT.RUNMODAL(PrintDocumentArguments.ReportID,TRUE,FALSE,RecordVariant);
```

Alternatives:

PAGE.RUNMODAL(PageID,RecordVariant)

CODEUNIT.RUN(CodeunitID,RecordVariant)

CalculateSalesDiscounts(RecordVariant) // Instead of Sales Header

Variant façade

Get the record ref

If support for RecordRef and RECORDID is needed
Assignment to Variant needed before usage – compiler limitation

```
PrintDocument(RecordVariant : Variant;PrintDocumentArguments : Record "Print Document Arguments")
DataTypeManagement.GetRecordRef(RecordVariant,RecRef);
RecordVariant := RecRef;

REPORT.RUNMODAL(PrintDocumentArguments.ReportID,TRUE,FALSE,RecordVariant);
```

Variant façade

Table Specific Code

```
PrintDocument(RecordVariant : Variant;PrintDocumentArguments : Record "Print Document Arguments")
DataTypeManagement.GetRecordRef(RecordVariant,RecRef);
RecordVariant := RecRef;

PrepareRecord(RecRef,PrintDocumentArguments,ReportSelectionUsage);
LOCAL PrepareRecord(RecordRef : RecordRef;
  PrintDocumentArguments : Record "Print Document Arguments";VAR ReportSelectionUsage : Option)
CASE RecordRef.NUMBER OF
  DATABASE::"Sales Header":
  BEGIN
    RecordRef.SETTABLE(SalesHeader);
    SalesHeader.CalcInvDiscForHeader;
    ReportSelectionUsage := SalesHeader.GetReportSelectionUsage;
  END;
  DATABASE::"Purchase Header":
  BEGIN
    RecordRef.SETTABLE(PurchaseHeader);
    PurchaseHeader.CalcInvDiscForHeader;
    ReportSelectionUsage := PurchaseHeader.GetReportSelectionUsage;
  END;
  DATABASE::"Bank Account Statement":
    ReportSelectionUsage := ReportSelections.Usage::"B.Stmt";
END;
```

Variant façade

Get the record back from the variant

SalesHeader := RecordVariant	**Removes filters!!!**
SalesHeader.Copy(RecordVariant)	**Correct!**
RecRef.SETTABLE(SalesHeader)	**Correct!**

Variant façade

Overview

- Clear separation between specific and common
- Single flow – removed duplication
- Minimized cost of adding a record
- Easy to extend with hooks and processes

```
PrintDocument(RecordVariant : Variant;PrintDocumentArguments : Record "Print Document Argument")
// Common Code Block
DataTypeManagement.GetRecordRef(RecordVariant,RecordRef); // Optional

// Specific Code
PrepareRecord(RecordRef,PrintDocumentArguments,ReportSelectionUsage);

// Common Code
ReportSelections.SETRANGE(Usage,ReportSelectionUsage);
RecordVariant := RecordRef;
REPORT.RUNMODAL(ReportSelections."Report ID",TRUE,FALSE,RecordVariant);
```

```
LOCAL PrepareRecord(VAR RecordRef : RecordRef;
  PrintDocumentArgument : Record "Print Document Argument";VAR ReportSelecitonUsage : Option)

CASE RecordRef.NUMBER OF
  DATABASE::"Sales Header":
    RecordRef.SETTABLE(SalesHeader);
    SalesCalcDisc(SalesHeader);
    ReportSelecitonUsage := SalesHeader.GetReportSelectionUsage;
  DATABASE::"Purchase Header":
    RecordRef.SETTABLE(PurchaseHeader);
    PurchaseHeader.CalculateDiscount;
    ReportSelecitonUsage := SalesHeader.GetReportSelectionUsage;
  DATABASE::"Bank Account Statement":
    ReportSelecitonUsage := ReportSelection.Usage::"B.Stmt";
  // And other tables here
  ELSE
    ERROR(STRSUBSTNO(UnSupportedTableTypeErr,FORMAT(DocumentRecordRef.NAME)));
END;
```

Variant façade

Consequences

- Don’t use it as a hammer! Not needed if there are only few tables
- Cannot pass variant with VAR – Need to fetch the record again
- Careful with filters – Test!
- Case statements can explode – use Argument Table or Rules Table
- Try to limit the number of case statements, avoid if possible

Suggestion – Right number of case statements:

0 – Best Number
1 – OK
2 – Please explain yourself
3 – No way!

The Data-driven Blocked Entity Pattern

The Data-driven Blocked Entity Pattern

Problem

Once created, a NAV record can be used for many things

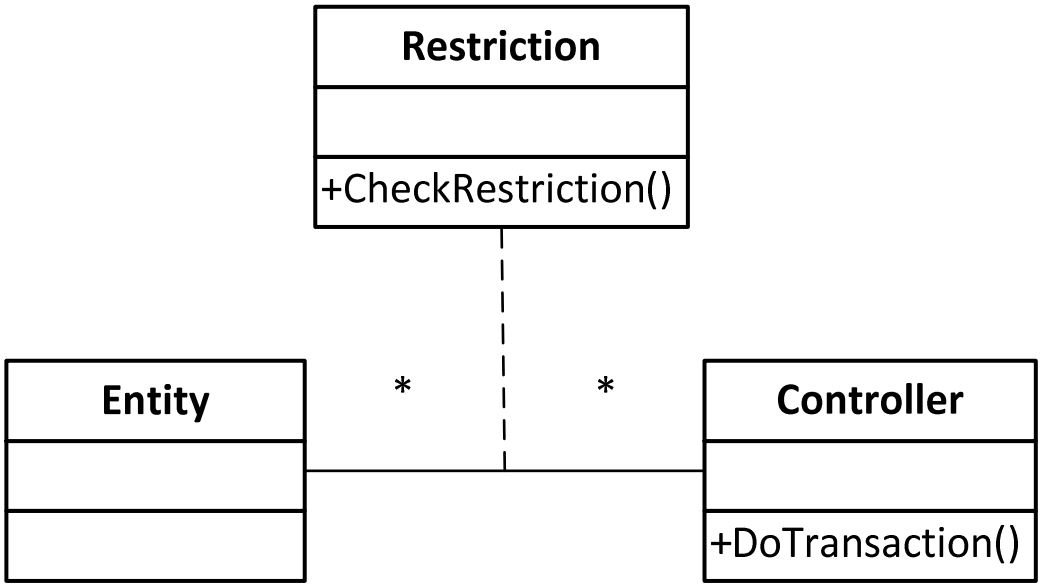
You might want to restrict its usage until a condition is met

Example: A new customer should not be used for posting until it is approved

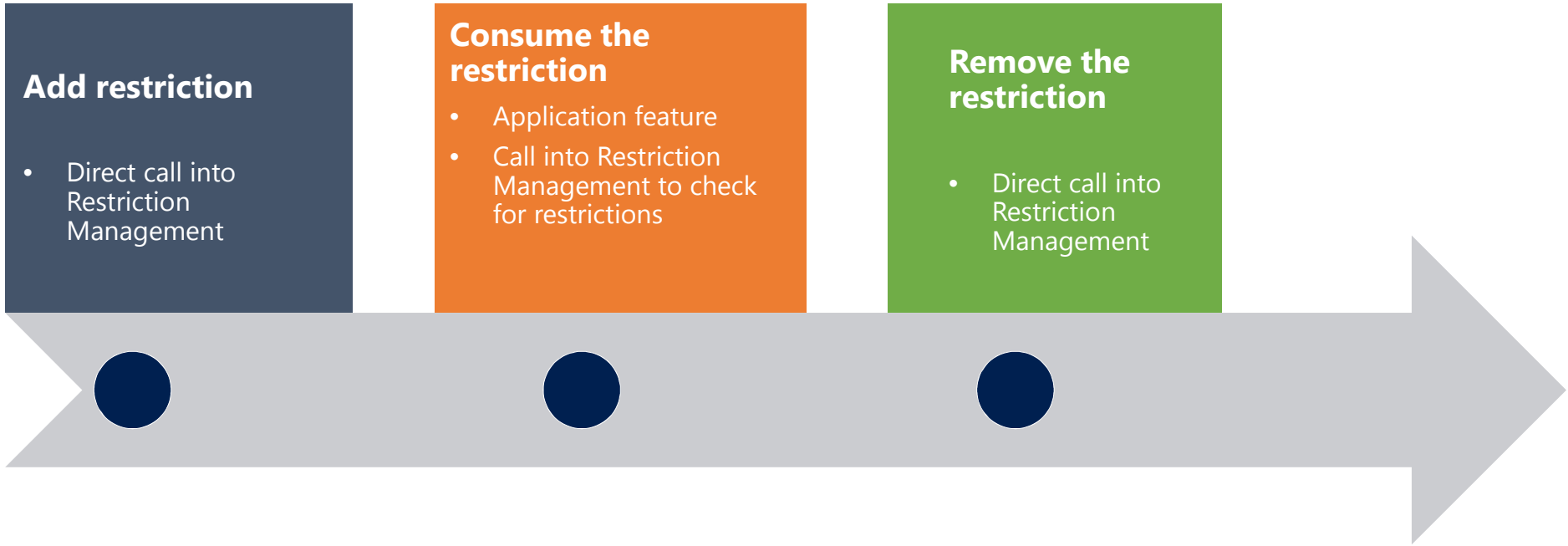
Solution

Use a generic mechanism for adding and lifting restrictions on a given record

The Data-driven Blocked Entity Pattern



The Data-driven Blocked Entity Pattern



The Data-driven Blocked Entity Pattern

Example

To restrict posting Gen. Journal Lines if a customer has not been added in Account No. field:

```
[EventSubscriber] RestrictGenJournalLineAfterInsert(VAR Rec : Record "Gen. Journal Line";RunTrigger : Boolean)
IF (Rec."Account Type" = Rec."Account Type"::Customer) AND (Rec."Account No." = '') THEN
  RestrictGenJournalLine(Rec);
```

```
[EventSubscriber] RemoveGenJournalLineRestrictionsAfterModify(VAR Rec : Record "Gen. Journal Line";VAR xRec : Record
"Gen. Journal Line";RunTrigger : Boolean)
IF NOT ((Rec."Account Type" = Rec."Account Type"::Customer) AND (Rec."Account No." <> '')) THEN
  AllowRecordUsage(Rec.RECORDID);
```

The Data-driven Blocked Entity Pattern

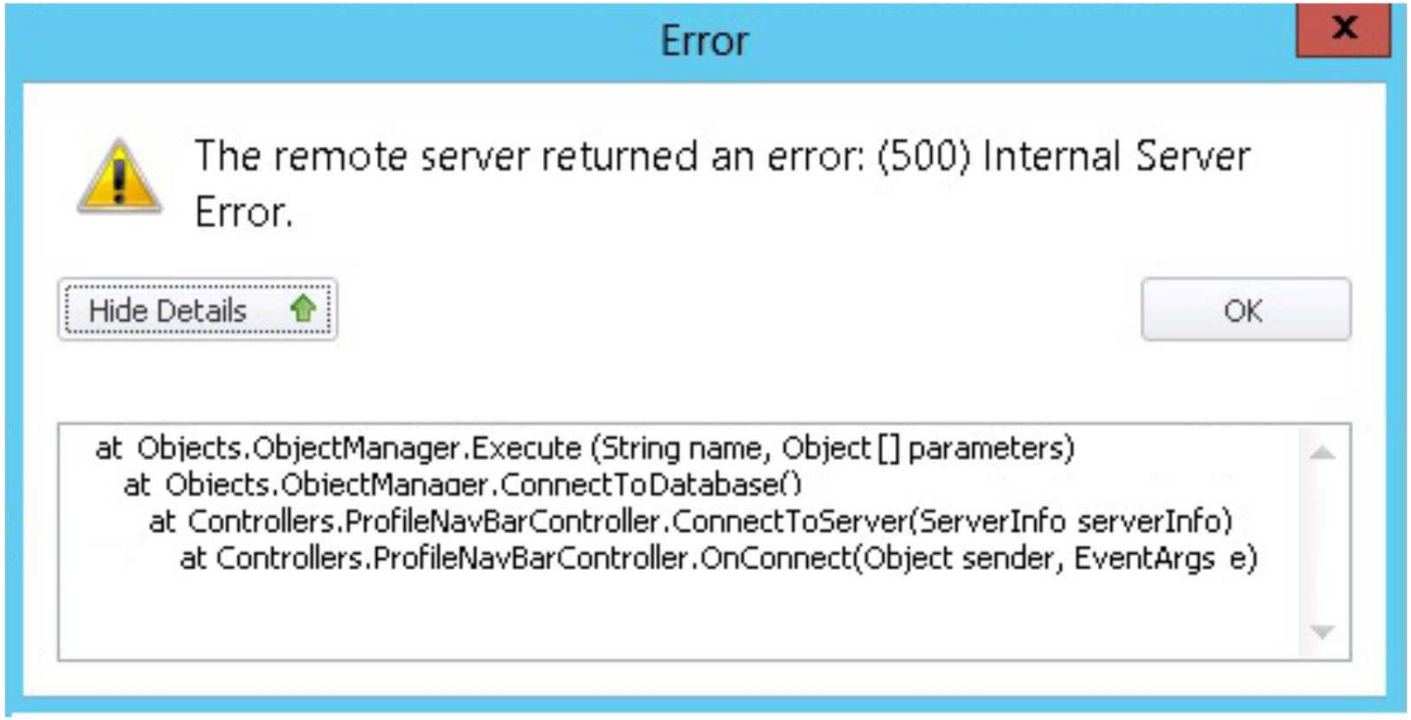
The Good

- Unlike the Blocked Entity pattern, no need for adding special fields in new tables
- Scalable and reusable
- The API is there already

The Bad

- You can only add a restriction once per record
- The restriction is global, their use cannot be refined

.NET Exception Handling – NAV 2016



.NET Exception Handling

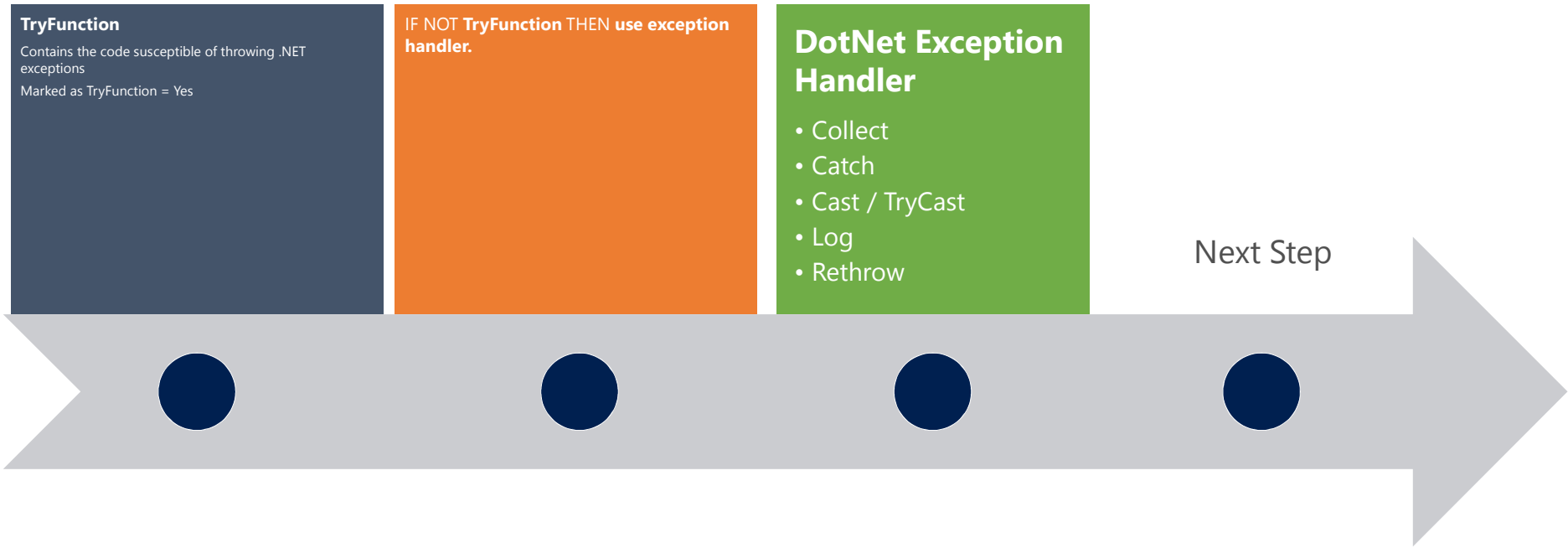
Problem

- Unhandled .NET exceptions are not actionable
- User does not understand what the problem is
- No Try-Catch clause in C/AL

Solution

- .NET exception handling in C/AL

.NET Exception Handling in NAV 2016



.NET Exception Handling in NAV 2016

Encapsulate the communication in a TryFunction:

```
[TryFunction]
PROCEDURE SendRequestToWebService@17();
VAR
  WebRequestHelper@1000 : Codeunit 1299;
  HttpWebRequest@1007 : DotNet "'System, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089'.System.Net.HttpWebRequest";
  HttpStatusCode@1002 : DotNet "'System, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089'.System.Net.HttpStatusCode";
  ResponseHeaders@1001 : DotNet "'System, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089'.System.Collections.Specialized.NameValueCollection";
  ResponseInStream@1006 : InStream;
BEGIN
  CheckGlobals;

  BuildWebRequest(GlobalURL,HttpWebRequest);

  ResponseInStreamTempBlob.INIT;
  ResponseInStreamTempBlob.Blob.CREATEINSTREAM(ResponseInStream);
  CreateSoapRequest(HttpWebRequest.GetRequestStream,GlobalRequestBodyInStream,GlobalUsername,GlobalPassword);


WebRequestHelper.GetWebResponse(HttpWebRequest,HttpWebResponse,ResponseInStream,HttpStatusCode,ResponseHeaders,GlobalProgressDialogEnabled)
;

  ExtractContentFromResponse(ResponseInStream,ResponseBodyTempBlob);
END;
```

.NET Exception Handling in NAV 2016

Wrap the call and handle the result

```
LOCAL PROCEDURE SendDataToConversionService@1(VAR PaymentFileTempBlob@1003 : Record 99008535;BodyTempBlob@1004 : Record
99008535;PostingExch@1007 : Record 1220);
VAR
  BankDataConvServiceSetup@1000 : Record 1260;
  WebServiceRequestMgt@1001 : Codeunit 1290;
  BodyInStream@1005 : InStream;
  ResponseInStream@1002 : InStream;
BEGIN
  IF NOT BodyTempBlob.Blob.HASVALUE THEN
    ERROR(NoRequestBodyErr);

  PrepareSOAPRequestBody(BodyTempBlob);

  COMMIT;

  BankDataConvServiceSetup.GET;
  BodyTempBlob.Blob.CREATEINSTREAM(BodyInStream);
  WebServiceRequestMgt.SetGlobals(BodyInStream,
    BankDataConvServiceSetup."Service URL",BankDataConvServiceSetup."User Name",BankDataConvServiceSetup.GetPassword);

  IF NOT WebServiceRequestMgt.SendRequestToWebService THEN
    WebServiceRequestMgt.ProcessFaultResponse;

  …
```

.NET Exception Handling in NAV 2016

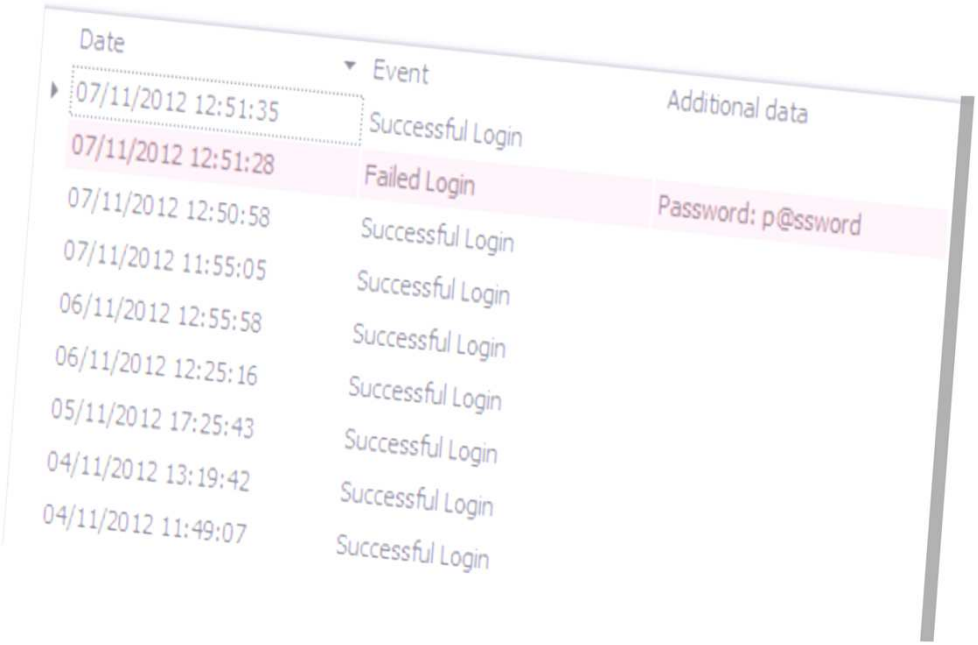
The Good

- Allows for user-friendly errors
- Reusable solution
- Works for both .NET and NAV errors

The Bad

- Not suitable for transactions

Activity Log



Date	Event	Additional data
07/11/2012 12:51:35	Successful Login	
07/11/2012 12:51:28	Failed Login	Password: p@ssword
07/11/2012 12:50:58	Successful Login	
07/11/2012 11:55:05	Successful Login	
06/11/2012 12:55:58	Successful Login	
06/11/2012 12:25:16	Successful Login	
05/11/2012 17:25:43	Successful Login	
04/11/2012 13:19:42	Successful Login	
04/11/2012 11:49:07	Successful Login	

Activity Log

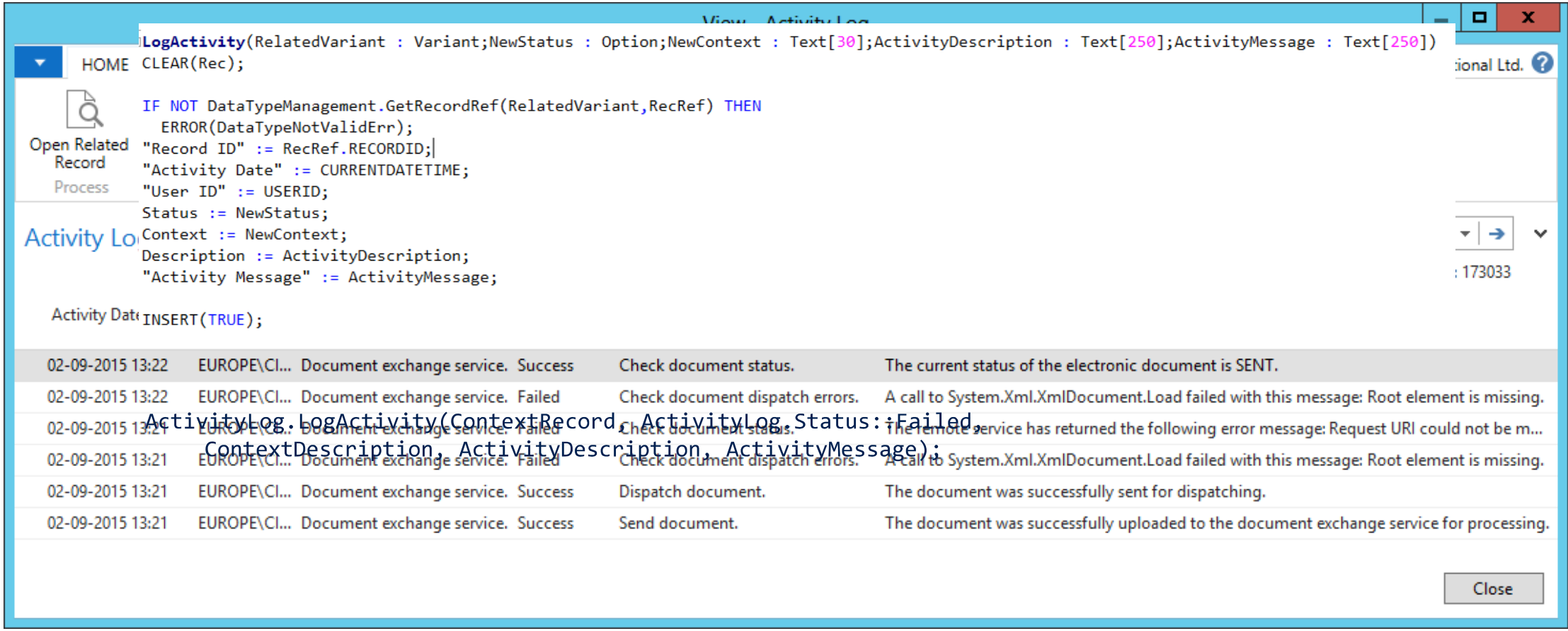
Problem statement

1. “What happened?”
2. “I cannot reproduce the problem”

Solution

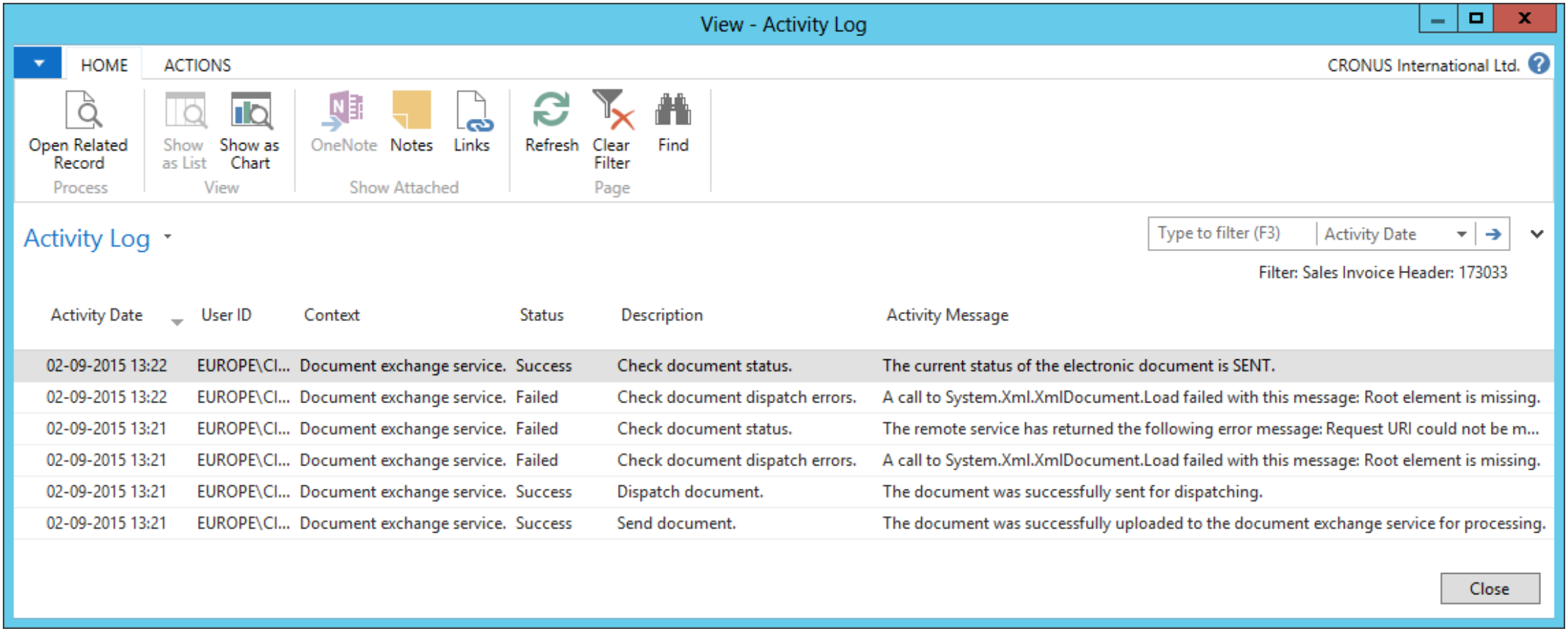
NAV Activity Log

Activity Log




```
ActivityLog.LogActivity(ContextRecord, ActivityLog.Status::Failed,
    ContextDescription, ActivityDescription, ActivityMessage);
```

PAG 710 Activity Log
TAB 710 Activity Log


View - Activity Log
HOME
ACTIONS
CRONUS International Ltd.
Open Related Record
Process
Show as List
Show as Chart
View
OneNote
Notes
Links
Show Attached
Refresh
Clear Filter
Find
Page
Activity Log
Type to filter (F3)
Activity Date
Filter: Sales Invoice Header: 173033
Activity Date
User ID
Context
Status
Description
Activity Message
02-09-2015 13:22 EUROPE\CI... Document exchange service. Success Check document status. The current status of the electronic document is SENT.
02-09-2015 13:22 EUROPE\CI... Document exchange service. Failed Check document dispatch errors. A call to System.Xml.XmlDocument.Load failed with this message: Root element is missing.
02-09-2015 13:21 EUROPE\CI... Document exchange service. Failed Check document status. The remote service has returned the following error message: Request URI could not be m...
02-09-2015 13:21 EUROPE\CI... Document exchange service. Failed Check document dispatch errors. A call to System.Xml.XmlDocument.Load failed with this message: Root element is missing.
02-09-2015 13:21 EUROPE\CI... Document exchange service. Success Dispatch document. The document was successfully sent for dispatching.
02-09-2015 13:21 EUROPE\CI... Document exchange service. Success Send document. The document was successfully uploaded to the document exchange service for processing.
Close

```
LogActivity(RelatedRecordID : RecordID;
            NewStatus : Option;NewContext : Text[30];
            ActivityDescription : Text[250];ActivityMessage : Text[250])

 CLEAR(Rec);

 "Record ID" := RelatedRecordID;
 "Activity Date" := CURRENTDATETIME;
 "User ID" := USERID;
 Status := NewStatus;
 Context := NewContext;
 Description := ActivityDescription;
 "Activity Message" := ActivityMessage;

 INSERT(TRUE);
```

```
ActivityLog.LogActivity
   (ContextRecordID, ActivityLog.Status::Failed, ContextDescription, ActivityDescription, ActivityMessage);
```

Activity log

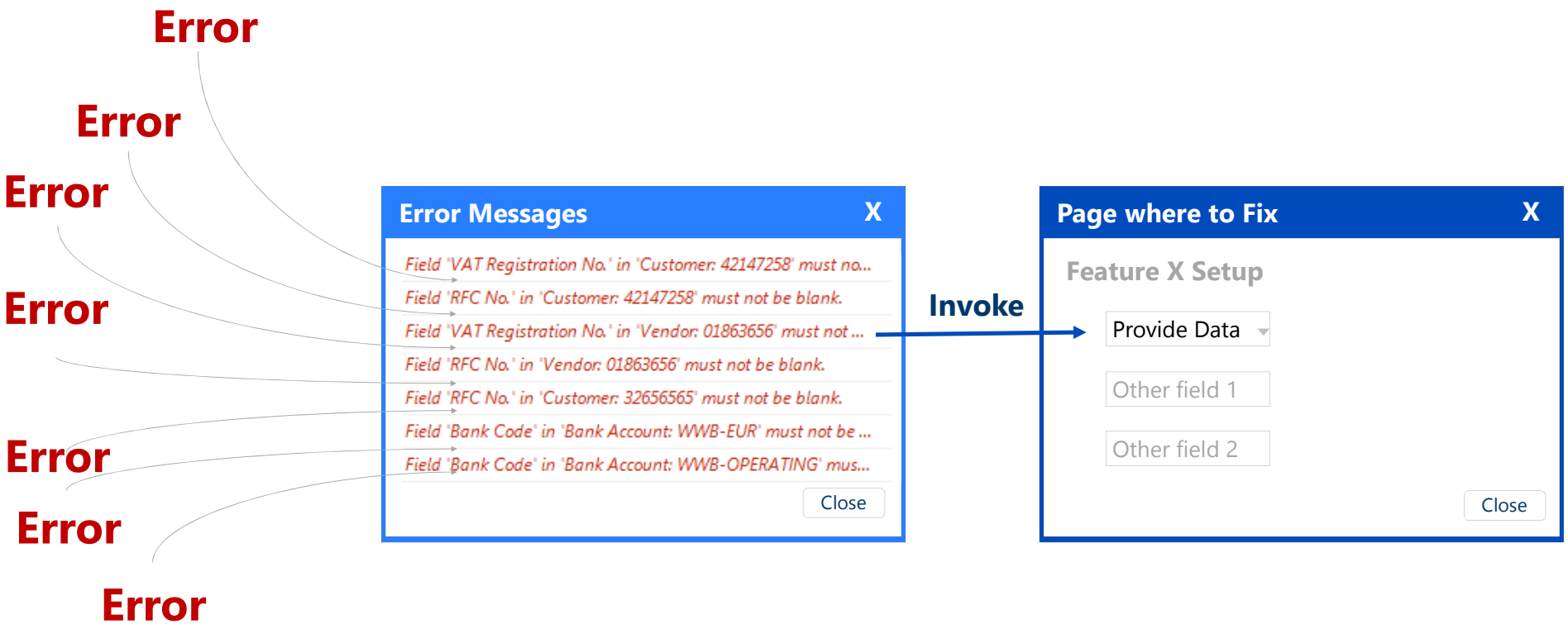
The Good

- Generic implementation, uses variants
- Out-of-the box API

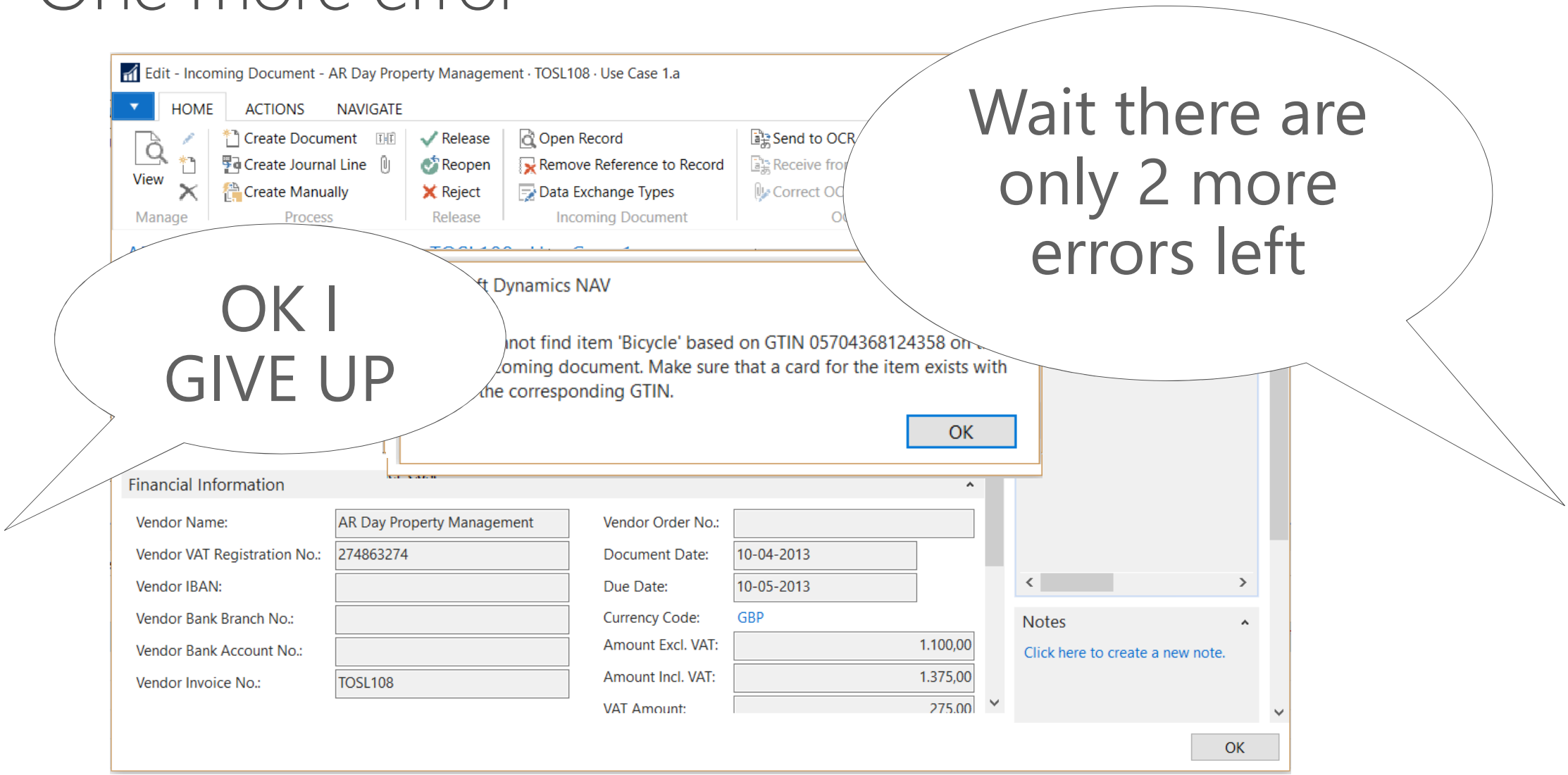
The Bad

- Cleanup required so table does not get huge with obsolete data

Error Message Processing



One more error



Error Message Processing

Problem

- One More Error is bad UX
- Test fields and error messages not very helpful
- Need common Error Processing

Solution

- Collect and show error messages in a single place
- Provide a link to a page where to fix it

Error Message Processing

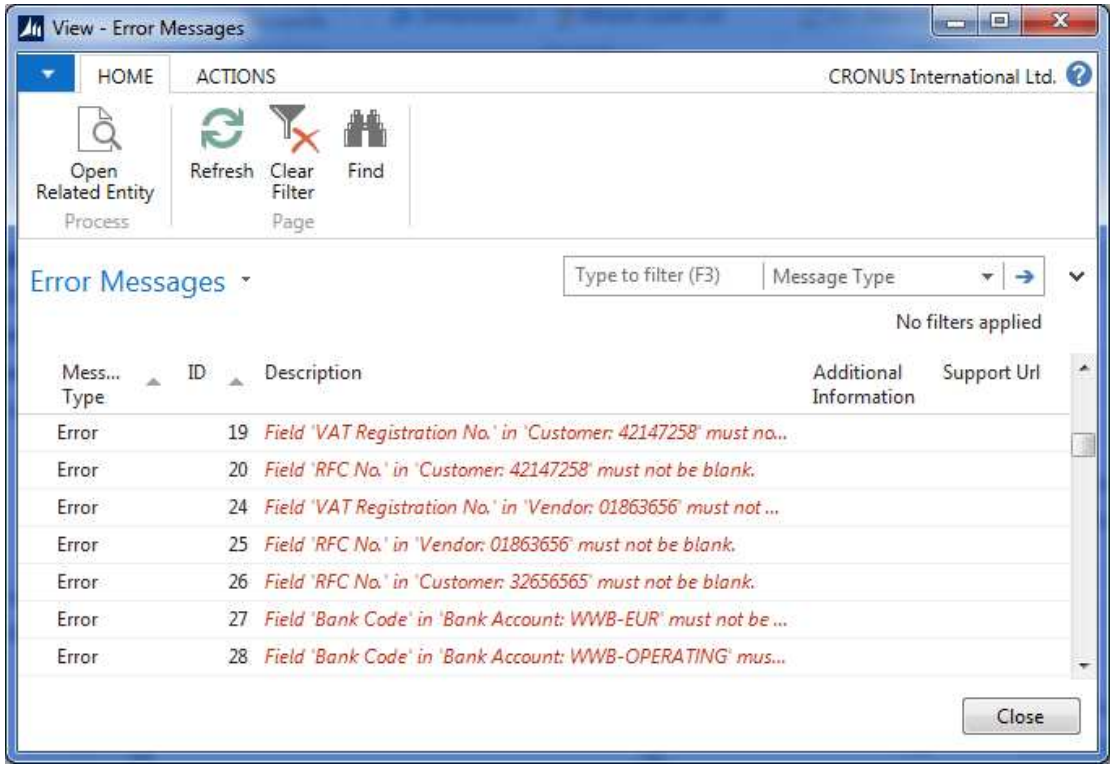
Usage

Logs 3 types of messages:

- Error
- Warning
- Information

Two possible implementations:

- Temporary table - short running process
- Persist error messages - long running / batch job



Error Message Processing

AR Day Property Management · TOSL108 · Use Case 1.a

General

Description:	Use Case 1.a	Status:	Failed
Main Attachment:	Use Case 1.a.xml	OCR Status:	
Record:		Job Queue Status:	
		Posted:	☐

Financial Information

Vendor Name:	AR Day Property Management	Vendor Order No.:	
Vendor VAT Registration No.:	274863274	Document Date:	10-04-2013
Vendor IBAN:		Due Date:	10-05-2013
Vendor Bank Branch No.:		Currency Code:	GBP
Vendor Bank Account No.:		Amount Excl. VAT:	1.100,00
Vendor Invoice No.:	TOSL108	Amount Incl. VAT:	1.375,00
		VAT Amount:	275,00

Errors and Warnings

Open Related Record | View Details | Find

Message Type	Field Name	Description
Error		*Cannot find an appropriate G/L account for the line with description 'Paper Subscription'. Choose the Map Text to Account button, and then map the co*
Error		*Cannot find a unit of measure with International Standard Code C62. Make sure that a unit of measure code exists with International Standard Code C6*
Error		*Cannot find an appropriate G/L account for the line with description 'Bicycle'. Choose the Map Text to Account button, and then map the core part of 'B*
Error		*Cannot find item 'Bicycle' based on GTIN 05704368124358 on the incoming document. Make sure that a card for the item exists with the correspondin*
Error		*Cannot find a unit of measure with International Standard Code C62. Make sure that a unit of measure code exists with International Standard Code C6*

Error Message Processing

Log Errors

```
LogIfEmpty(RecRelatedVaric
LogIfLengthExceeded(RecRel
LogIfInvalidCharacters(Rec
LogIfOutsideRange(RecRelat
LogIfGreaterThan(RecRelate
LogIfLessThan(RecRelatedVa
LogIfEqualTo(RecRelatedVar
LogIfNotEqualTo(RecRelatec
LogSimpleMessage(MessageTy
LogMessage(RecRelatedVaric
LogSimpleMessageInContext(
LogMessageInContext(Contex
LogDetailedMessage(RecRela
AddMessageDetails(Messagel
SetContext(MessageID : Int
```

Check for Errors

```
HasErrorMessagesRelatedTo(R
ErrorMessageCount(LowestSev
HasErrors(ShowMessage : Boo
ErrorMessageCountInContext(
HasErrorsInContext(ContextR
```

Show Errors

```
ShowErrorMessages(RollBackOnE
ShowErrorMessagesInContext(Co
```

The entire code is in:

- Table 700 – Error Message
- Page 700 – Error Messages
- Page 701 – Error Messages Part

Link to page

- Cod 700 – Page Management
- Cod 701 – Data Type Management

Error Message Processing

The Good

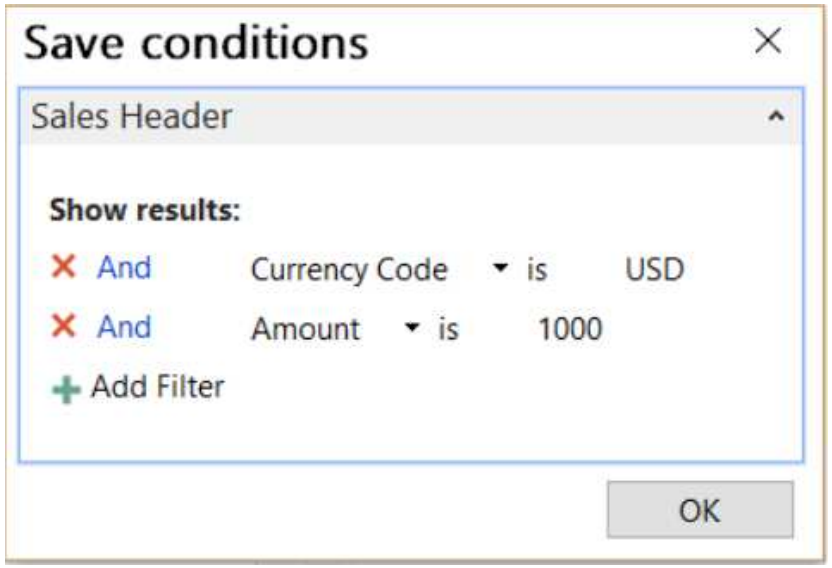
- Generic implementation, uses recordid
- Out-of-the box helpers

The Bad

- No option to copy/parse the error messages
- Cleanup required (for the persistent solution)
- Tricky code to persist in case of errors

Dynamic Request Page

```
ShowRequestPage(SalesHeader);
```



Dynamic Request Page

Problem

- Enable users to specify conditions (e.g. when to use a record)
- Need a specific page or a report for each table

Solution

- Provide a generic request page for a given table
- No additional objects needed – page is build dynamically
- [Optional] Store filter as a blob in database

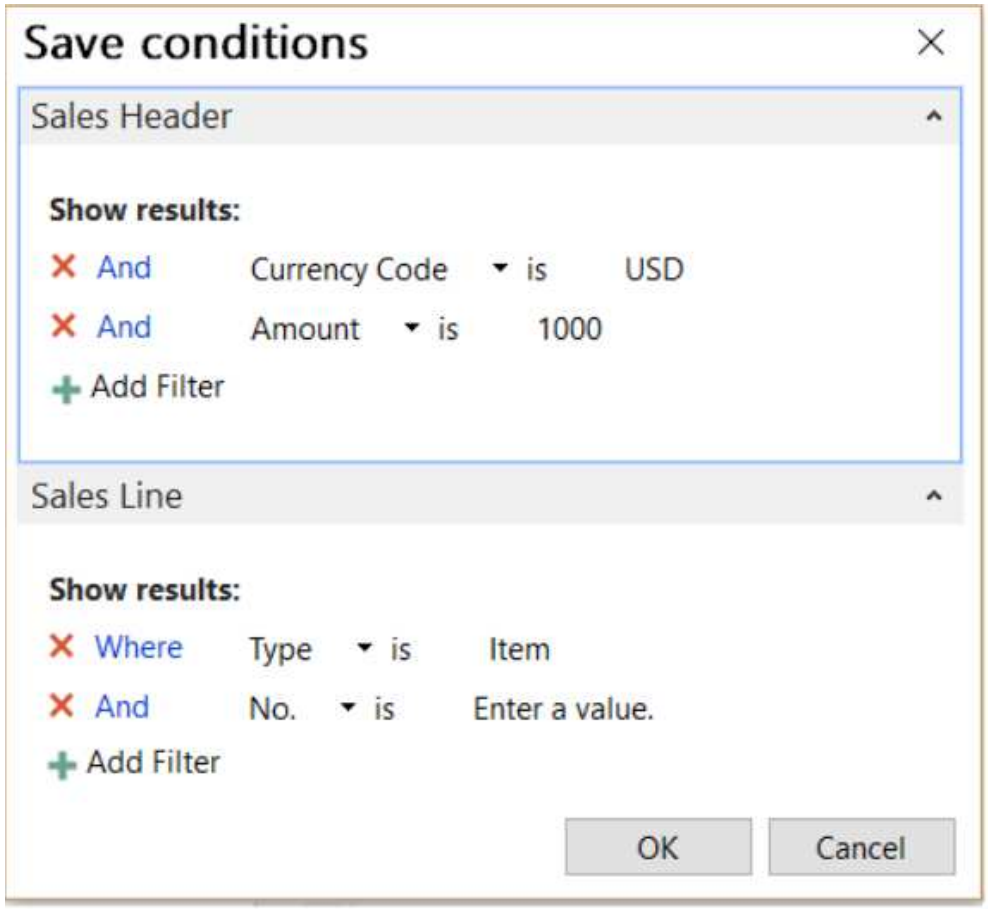
Build a request page from code

```
CollectConditions() : Text
FilterPageBuilder.PAGECAPTION := 'Email Profiles';

FilterPageBuilder.ADDTABLE(SalesHeaderCaptionTxt,DATABASE::"Sales Header");
FilterPageBuilder.ADDFIELD(SalesHeaderCaptionTxt,SalesHeader.Amount);
FilterPageBuilder.ADDFIELD(SalesHeaderCaptionTxt,SalesHeader."Currency Code");

IF FilterPageBuilder.RUNMODAL THEN
  EXIT(FilterPageBuilder.GETVIEW(CaptionTxt));
```

Dynamic Request Page – Multiple records



Dynamic Request Page – Multiple records

Dynamic Request Page Entities

Name	Description	Table ID	Table Caption	Related Table ID	Related Table Caption
INCOMINGD...	Incoming Document	130	Incoming Document	133	Incoming Document Attachment
INCOMINGD...	Incoming Document	133	Incoming Document Attachment	130	Incoming Document
PURCHDOC	Purchase Document	38	Purchase Header	39	Purchase Line
SALESDOC	Sales Document	36	Sales Header	37	Sales Line

Dynamic Request Page Fields

Table ID	Table Caption	Field ID	Field Caption
36	Sales Header	2	Sell-to Customer No.
36	Sales Header	23	Payment Terms Code
36	Sales Header	32	Currency Code
36	Sales Header	60	Amount
37	Sales Line	5	Type
37	Sales Line	6	No.
37	Sales Line	15	Quantity
37	Sales Line	100	Unit Cost

Entity

```
LOCAL RunDynamicRequestPage(VAR ReturnFilters : Text;Filters : Text) : Boolean
IF NOT TableMetadata.GET("Table ID") THEN
  EXIT(FALSE);

IF NOT RequestPageParametersHelper.BuildDynamicRequestPage(FilterPageBuilder,"Dynamic Req. Page Entity Name","Table ID") THEN
  EXIT(FALSE);

IF Filters <> '' THEN
  IF NOT RequestPageParametersHelper.SetViewOnDynamicRequestPage(
       FilterPageBuilder,Filters,"Dynamic Req. Page Entity Name","Table ID")
  THEN
    EXIT(FALSE);

FilterPageBuilder.PAGECAPTION := STRSUBSTNO(EventConditionsCaptionTxt,Description);
IF NOT FilterPageBuilder.RUNMODAL THEN
  EXIT(FALSE);

ReturnFilters :=
  RequestPageParametersHelper.GetViewFromDynamicRequestPage(FilterPageBuilder,"Dynamic Req. Page Entity Name","Table ID");

EXIT(TRUE);
```

COD 1530 - Request Page Parameters Helper

New Data Type – FilterPageBuilder

[Optional] Store results in BLOB fields

Events

- Concepts
- End-to-end Coding Example
- Why do we need Events?
- When do we need Events?
- How can YOU leverage Events in your verticals?

Events

Problem

- Short update cycles
- Continuous updates are key to SaaS
- Traditionally C/AL code is tightly coupled
- Modifications tend to have a large footprint in the standard application

Solution

- Enable loose coupling of methods
- Allows functional enhancement without touching standard objects
- Decrease the footprint
- Make updates safer and more simple
- Enable painless continuous updates
- Facilitate collaboration

Events

You can leverage Events to implement additional business processes without needing to touch standard objects at all.


DEMO

Extensibility Event Pattern

- Design a business process in a way that it becomes a framework for decoupled methods
- Wrap the process in publisher events:
  - OnBeforeProcess (➔ Publisher Event)
  - MainProcess (Executable Method)
  - OnAfterProcess (➔ Publisher Event)
- Enhance your business process by adding methods as Subscriber Events

The term “Process” is just used as a placeholder for the actual feature that you want to implement.

Extensibility Hook Pattern

- Design a business process in a way that it becomes a framework for decoupled methods
- Wrap the process into hooks:
  - OnBeforeProcess (➔ Call to relevant hooks)
  - MainProcess (Executable Method)
  - OnAfterProcess (➔ Call to relevant hooks)
- Enhance your business process by adding methods as codeunits that you can run with a record parameter
- Include new methods in your hook list

Extensibility Patterns

- Always: Keep track of whether your business process has been handled by an additional method (subscriber or hook):
  - EventHandled variable
  - “Event Handled” field
- “Event Pattern” and “Hook Pattern” solve the same problem
- “Event Pattern” depends on platform availability
- “Event Pattern” is preferred if available

Event Types

- Global Events (Codeunit 1)
- Trigger Events (Table / Page)
- Business Events (Code Declared Events)
- Integration Events (Code Declared Events)

Global Events

- Codeunit 1
- Declared as local Integration Events
- Currently no real system events, but a variation of Integration Events

Trigger Events - Table

- OnBeforeDeleteEvent
- OnAfterDeleteEvent
- OnBeforeInsertEvent
- OnAfterInsertEvent
- OnBeforeModifyEvent
- OnAfterModifyEvent
- OnBeforeRenameEvent
- OnAfterRenameEvent
- OnBeforeValidateEvent
- OnAfterValidateEvent

Order of Event Execution

- Table Trigger Before Event (e.g. OnBeforeDeleteEvent)
- Table Trigger (e.g. OnDelete)
- Global Trigger in Codeunit 1 (e.g. OnDatabaseDelete)
- *Database Operation (e.g. Record gets deleted)*
- Table Trigger After Event (e.g. OnAfterDeleteEvent)

Trigger Events - Page

- OnOpenPageEvent
- OnAfterGetRecordEvent
- OnAfterGetCurrRecordEvent
- OnNewRecordEvent
- OnInsertRecordEvent
- OnModifyRecordEvent
- OnDeleteRecordEvent
- OnBeforeValidateEvent
- OnAfterValidateEvent
- OnBeforeActionEvent
- OnAfterActionEvent
- OnQueryClosePageEvent
- OnClosePageEvent

Code Declared Event Publishers

- Business Events / Integration Events
- All Events are declared as functions
- Events cannot have code, only comments
- Events cannot have return values

- Best Practices:
  - Publish a local function on the object where it is raised
  - Only raise an event once
  - Stick meticulously to naming conventions

Event Subscribers

- You can only subscribe to events in codeunits
- Each event publisher can have multiple subscribers
- The order in which multiple subscribers invoke a publisher event cannot be defined and can vary from instance to instance
- Arguments passed from publisher to subscriber are bound by name and type
- The order of the arguments is irrelevant
- The number of arguments is irrelevant: only use the arguments you really need

Concepts: Business vs. Integration Events

- Behavior is almost identical; the difference is a conceptual one

- Integration Events are helpers to avoid code modification in standard objects
- Integration Events may be subject to change over time

- Business Events should not change
- Business Events define a formal contract
- Business Events constitute a public API
- Currently no Business Events in Dynamics NAV

Business Events

- When to introduce Business Events into your vertical
- Break your complete solution down into encapsulated methods
- Each method is a codeunit of its own with only one global function
- Each method is only ever called from the class (table) it belongs to
- Define which methods belong to which business processes
- Use Business Events to create an API that gives external access to your processes

A Business Event API

- Your Business Event API allows you to differentiate between types of developers / development
- Core development which changes or enhances your structure / architecture
  - Senior product developers
- Additional development which is limited to the defined contracts and interface points
  - External developers
  - Junior project / tenant extensibility developers

Where Used

- Dynamics CRM Integration
- Workflow
- Permission Recorder
- Custom Events for Posting
- Registering Service Connection
- NAV Extensions

Summary

1. Events

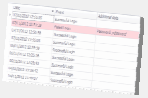
2. Passwords and sensitive data

3. Variant façade

4. Data-driven blocked entity

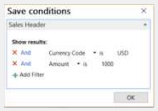
5. Try method

6. Logging



7. Error message processing

8. Dynamic request pages



Learn more

https://community.dynamics.com/nav/w/designpatterns

Think NAV
in design patterns.

Any Questions?

THANK YOU

WHEN YOU ARE PASSIONATE ABOUT MICROSOFT DYNAMICS NAV | *www.navtechdays.com*

NAV TECH DAYS 2015

mibuso.com

LUNCH BREAK

see you back in 60 min.