Consultingwerk
software architecture and development

Refactoring OpenEdge TTY or GUI Applications refactoring for Modern Web Applications

Mike Fechner
mike.fechner@consultingwerk.de

Mike Fechner

35 years of experience in Progress (from Version 5 to OpenEdge 12)

Active member of the OpenEdge community and speaker at international conferences

Expert in software architecture, object-oriented design, and web technologies

Who are We?

- Independent IT consulting company **specializing in OpenEdge** and complementary technologies
- **Global presence** with headquarter in Cologne and offices in the UK, USA, and Romania
- **Serving clients across Europe, North America, Australia, and South Africa**
- Provider of **advanced developer tools** and customized consulting services
- **In-depth expertise** in: GUI development with .NET and Angular, object-oriented programming and software architecture, application integration, and enterprise systems design
- Experts in **modernizing legacy OpenEdge applications**

Consultingwerk
software architecture and development

Modernization in Focus

Modernization of Legacy OpenEdge Applications

Deep Technical Expertise

Global IT Partner with Local Presence

More than Consulting – We Deliver Tools & Solutions

Agenda

- **Modernization Process**
- Application Architecture
- Dealing with (GLOBAL) SHARED Variables
- Dealing with messages or prompts
- Proparse
- Record Locking

Modernization drivers

- The obvious: a new user interface
  - Web interface
  - Modern desktop UI
  - Mobile or satellite applications

- Functional requirements
  - Integration with 3rd party applications (in and out)
  - Localization
  - Hard to keep up with new features
  - Redundancy and spaghetti code killing agility and maintainability

Modernization drivers

- Improved code quality / maintainability
  - Improvements to application longevity
  - Component independency
  - Module independence
  - Method length
  - Test driven development to improve quality and agility
- Get ready for a new generation of software developers
  - Foreseeable retirement of key developers
  - Need to make application attractive to young developers
  - Enable application for distributed development

Other modernization drivers

- OpenEdge Version upgrades
  - WebSpeed retired with OpenEdge 11.7 on April 1st 2025
  - Progress Dynamics not available in OpenEdge 12
  - Printing solutions, still anyone using Report Builder?

- AppServer enabling to improve performance
  - This is still a thing!
  - Customers still running large processing routines via Client/Server in Wifi

Modernization drivers

- Modernization drivers need agreement between all stake-holders
  - development team
  - business
- When time-pressure comes, goals not directly visible to end users may otherwise be sacrificed
  - code-optimization
  - adherence to architectural standards
  - test-driven-design
  - technical documentation

Know what will remain constant

- I expect that OpenEdge and PASOE will still be around 10 years from now
- I expect that OpenEdge will keep fundamentally backwards compatible with todays source-code
- Majority of application functionality should be moved to PASOE

- I will not even try to foresee the trends in user-interface technology in the next few years

Consultingwerk
software architecture and development

Quality of the application

- Are parts of the application reusable?
  - With no or little changes
  - Are major functional changes required?
  - Are major changes to the database structure required?

- Can parts of the application serve to describe the requirements
  - Legacy code review as part of the requirements definition
  - **Is the existing source code the only (complete) description of the application functionality?**

Development Team Skills

- New development process (agile)
- New tools (VS Code, Progress Developer Studio, SCM, Unit Tests, DevOps, Docker, Frontend tools) and Frameworks
- New architecture: Distributed
- New development languages
  - OOABL
  - HTML, JavaScript, TypeScript, rapidly changing
  - Desktop technologies
  - Web and Mobile frameworks

Agenda

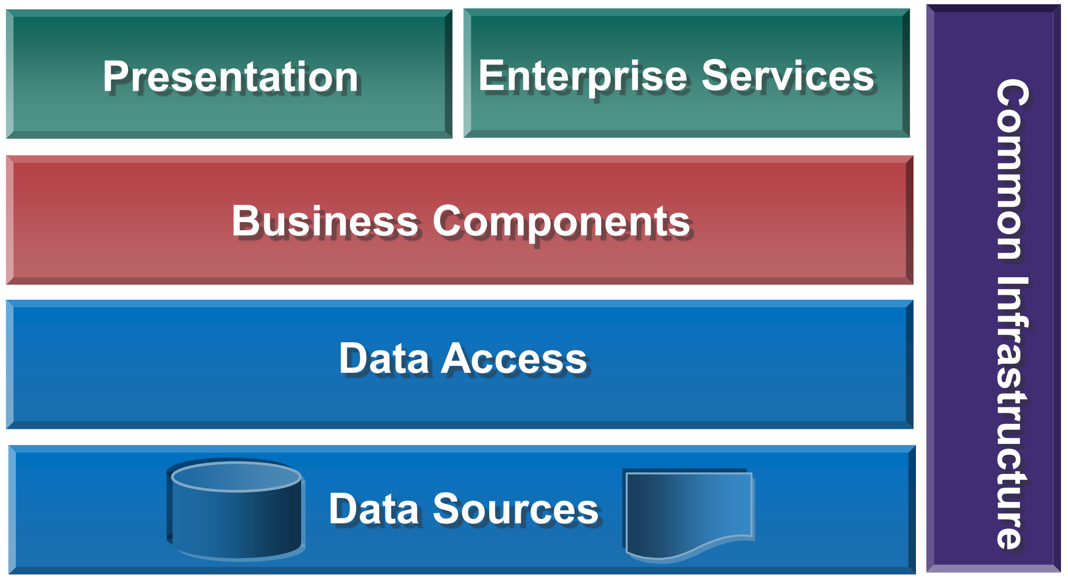
- Modernization Process
- **Application Architecture**
- Dealing with (GLOBAL) SHARED Variables
- Dealing with messages or prompts
- Proparse
- Record Locking

Consultingwerk
software architecture and development

The OpenEdge Reference Architecture (OERA)

The OpenEdge Reference Architecture (OERA) defines the general functional categories of components that comprise an application. It can be used as a high-level blueprint for developing OpenEdge service-oriented business applications.

Each layer of the OERA consists of distinct components, each with specific characteristics, roles and responsibilities. In addition, the OERA provides guidelines as to how each of the architectural components interacts. The following diagram illustrates the component architecture and the relationships between each of the components.



https://community.progress.com/s/question/0D54Q0000819wkqSAA/introduction-to-the-openedge-reference-architecture

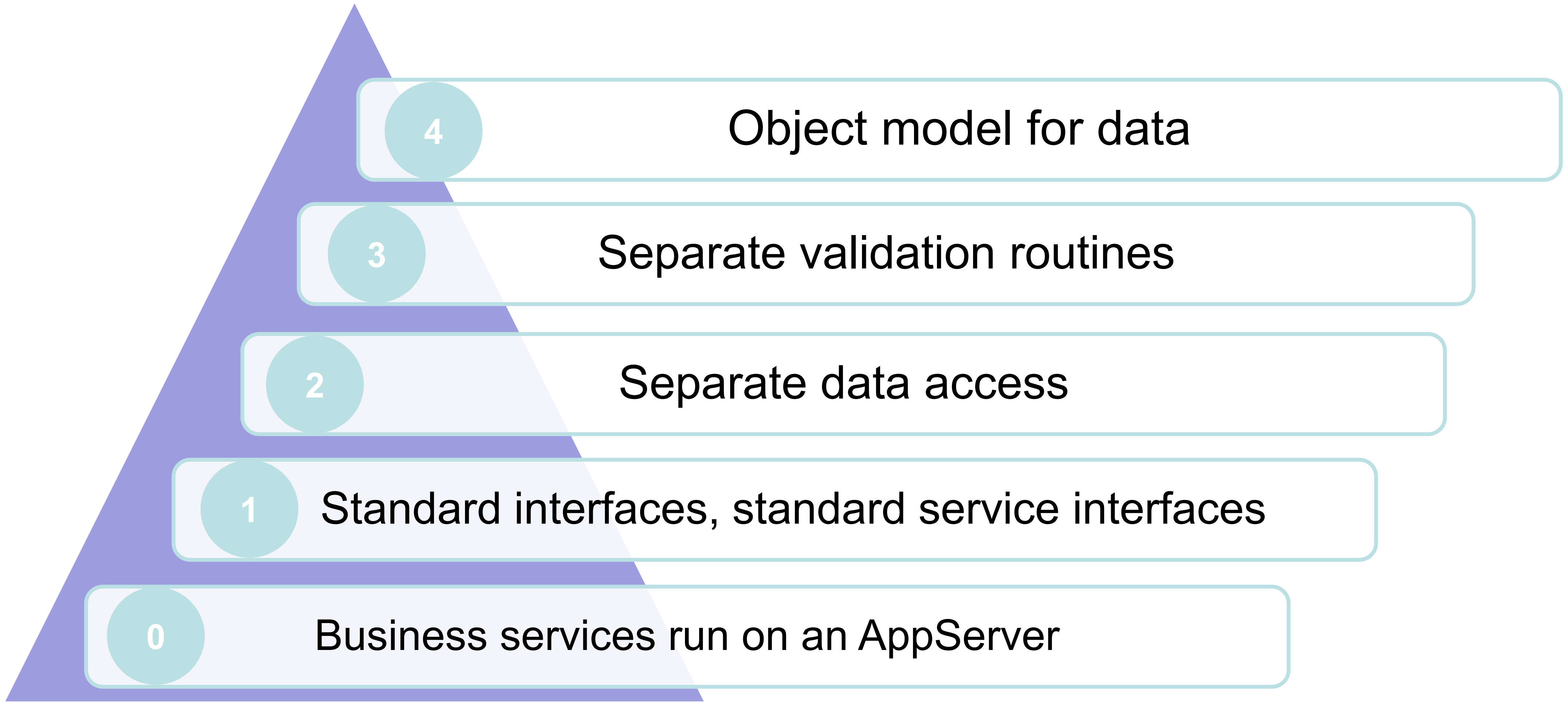
The OpenEdge Reference Architecture (OERA)

- Focus is on high-level architecture "blueprint"
- OERA is not prescriptive …
  - Choose to use procedural or OOABL code
  - Choose to implement some or all layers
  - Choose to keep existing code
- Service Interface Layer almost entirely ignored
- No guidance given on implementation, other than sample code

The Consultingwerk OERA Maturity Model

- An *opinionated* attempt to give application architects and developers some orientation in how to implement OERA compliant ABL applications
  - E.g. We don't believe a Data Source layer provides value
- Assumes that different developer teams have different requirements and expectations for the architecture and coding style of modernized ABL applications
- Builds upon the OERA and is focused on *implementation*, primarily relating to the Business Components and Data Access layers
- Soon on https://www.consultingwerk.com/news/blog

The Consultingwerk OERA Maturity Model


4
Object model for data
3
Separate validation routines
2
Separate data access
1
Standard interfaces, standard service interfaces
0
Business services run on an AppServer

Reduced reuse of legacy code?

- Implementing full separation of concerns can mean it's much harder to reuse existing blocks of code
  - Reusing large parts of existing code promises faster migration process
  - Existing unit- and system tests can continue to be used when reusing legacy code

- Find the balance between migration speed and risk reduction, and future-proofing and increased maintainability

Service Interface(s)

Bouncer

Babelfish

Service Interface(s)

- The Service Interface receives calls from clients or external consumers
- A very important and often under-appreciated component
  - The Service Interface is responsible for Validating the request (including Authentication and Authorization)
  - Ensuring the User-Session is in the correct state
  - Allocating the service (the Application Service, Business Task or Entity)
  - Converting the request data from an external format to internal
  - Converting the response data from internal format to external

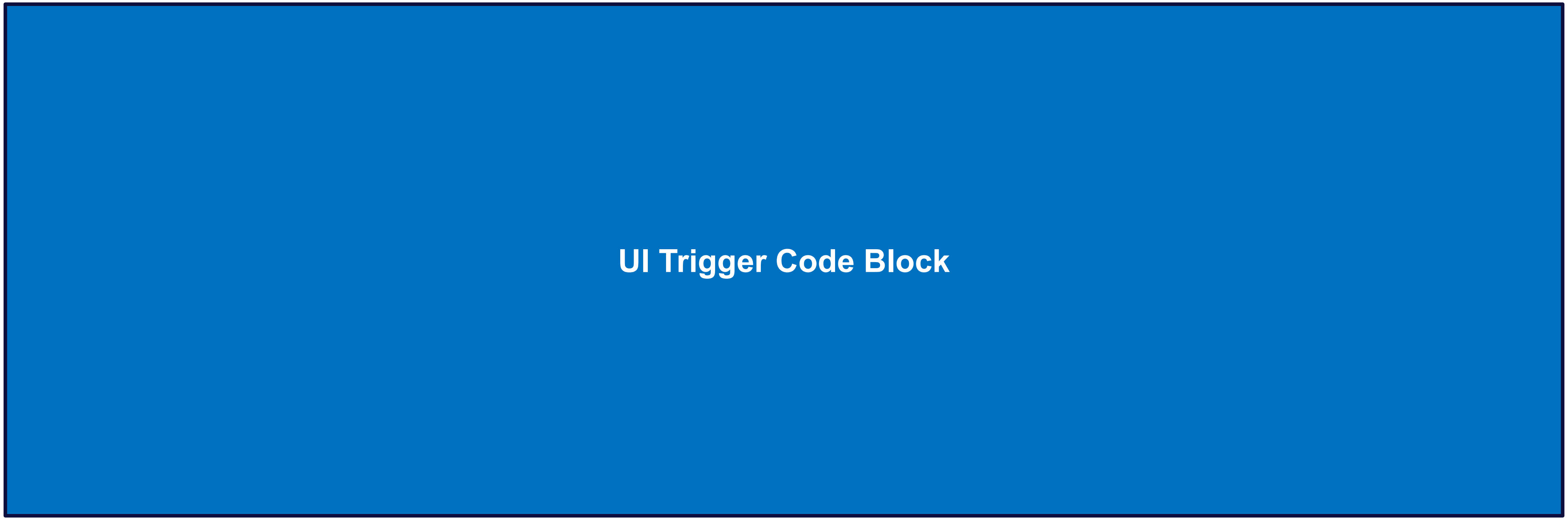
Service Interface(s)

- Business Logic is the most valuable piece of the application
- User interfaces come and go (TTY, ABL GUI, GUI for .NET, Web, Mobile, Chat, …)
- We do not want to rewrite – or even change the Business Logic for every new UI trend
- Multiple parallel used UI technologies should be using the same Business Logic
  - When there are very specific requirements for a single UI (e.g. Wizard style vs. plain data entry, consider using Application Service for this as an aggregate of multiple Business Tasks or Entities)

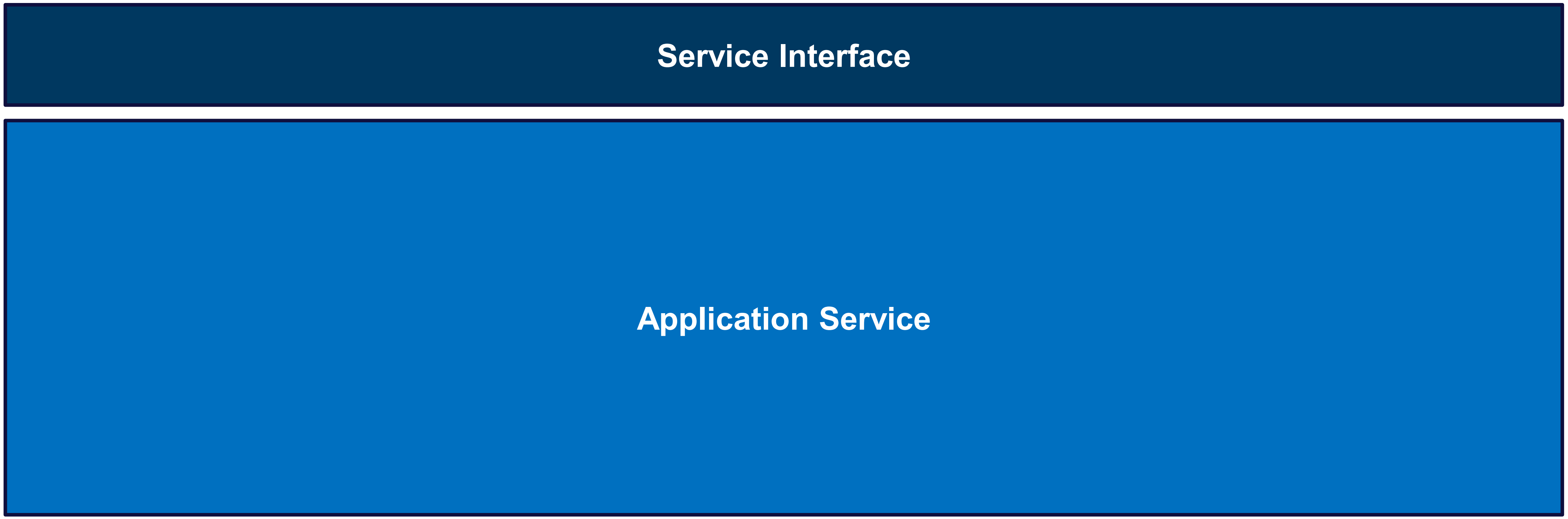
Top-down code-generalization

- Existing code considered to be closest to an application service
- First step is moving code from UI into an application service
- Simplifies automation during code-refactoring (almost statement by statement replaces)
- Further steps will improve code-reuse and single-concern by extracting code from application service into domain services
- Code de-duplication requires more design and guidelines
  - Into how many pieces do we cut the monolith?

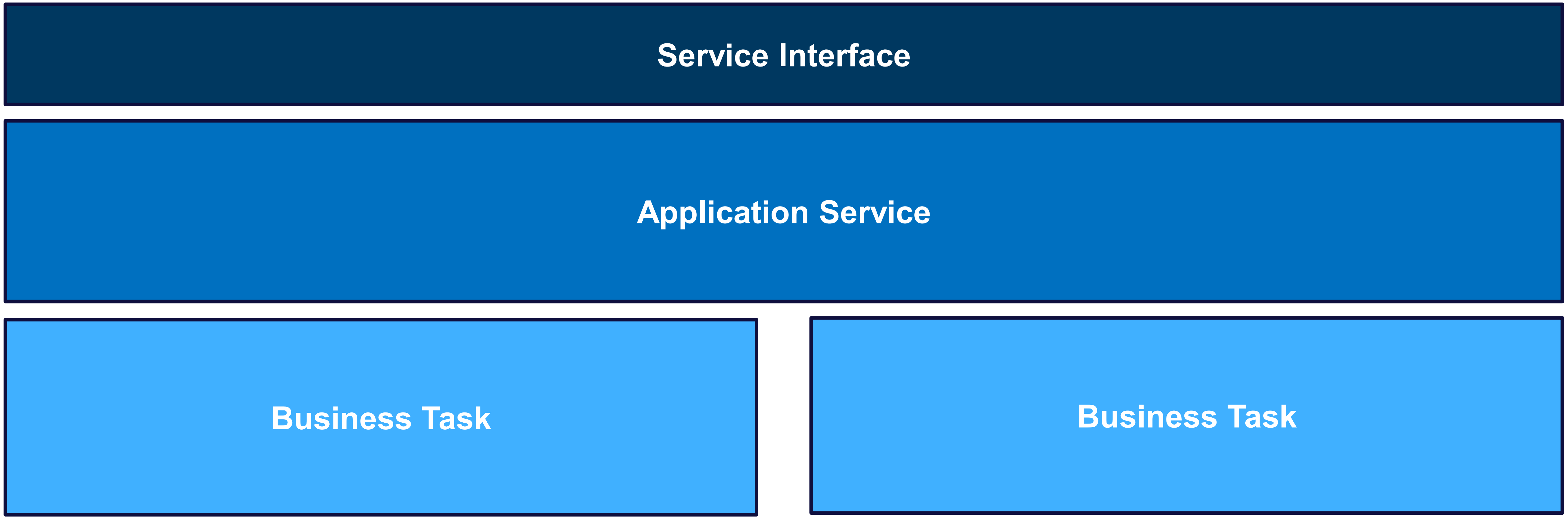
Top-down code-generalization



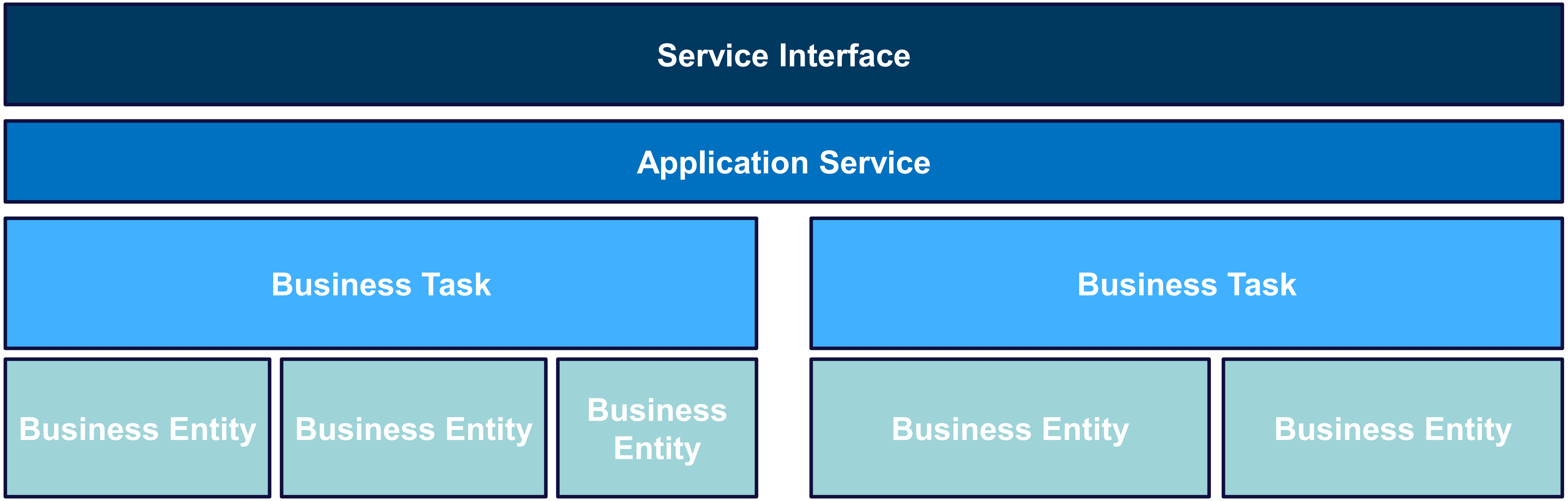
Top-down code-generalization



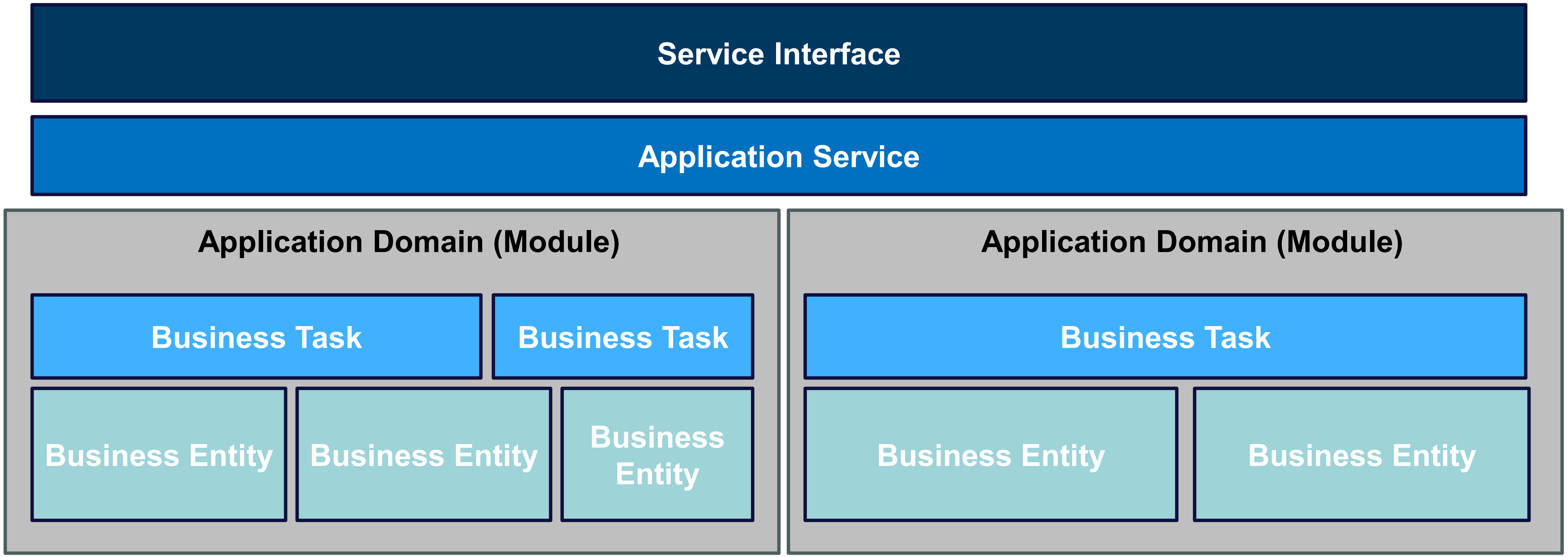
Top-down code-generalization



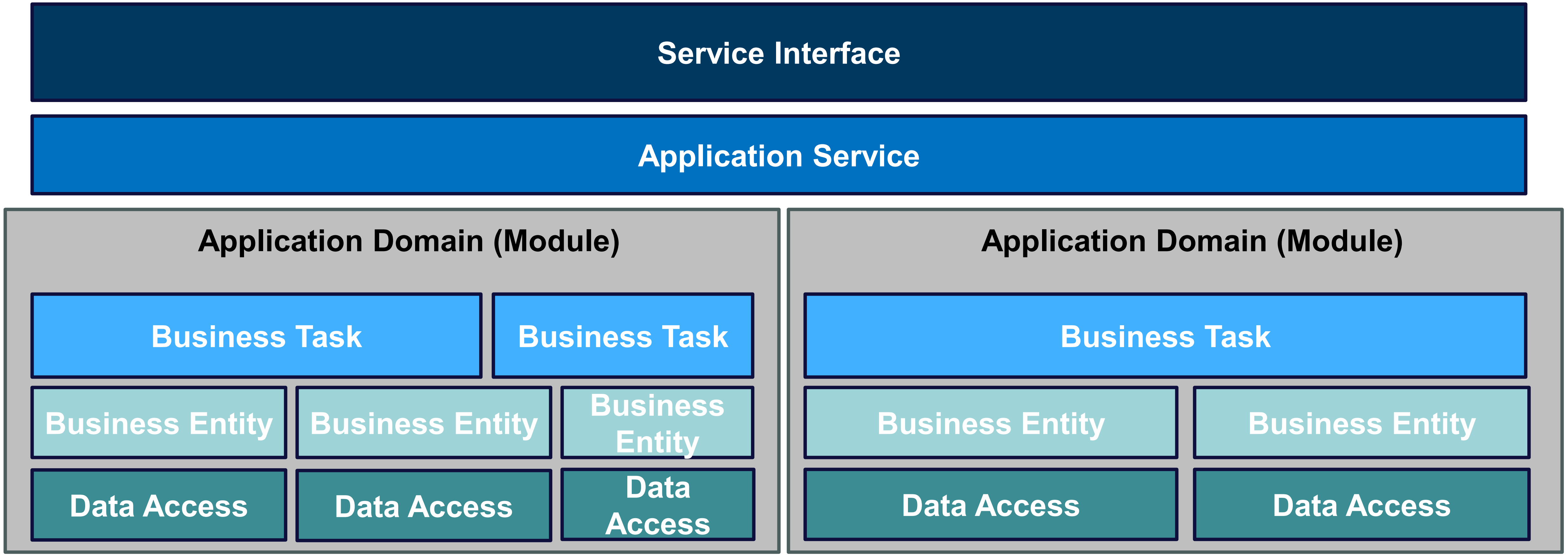
Top-down code-generalization



Top-down code-generalization



Top-down code-generalization

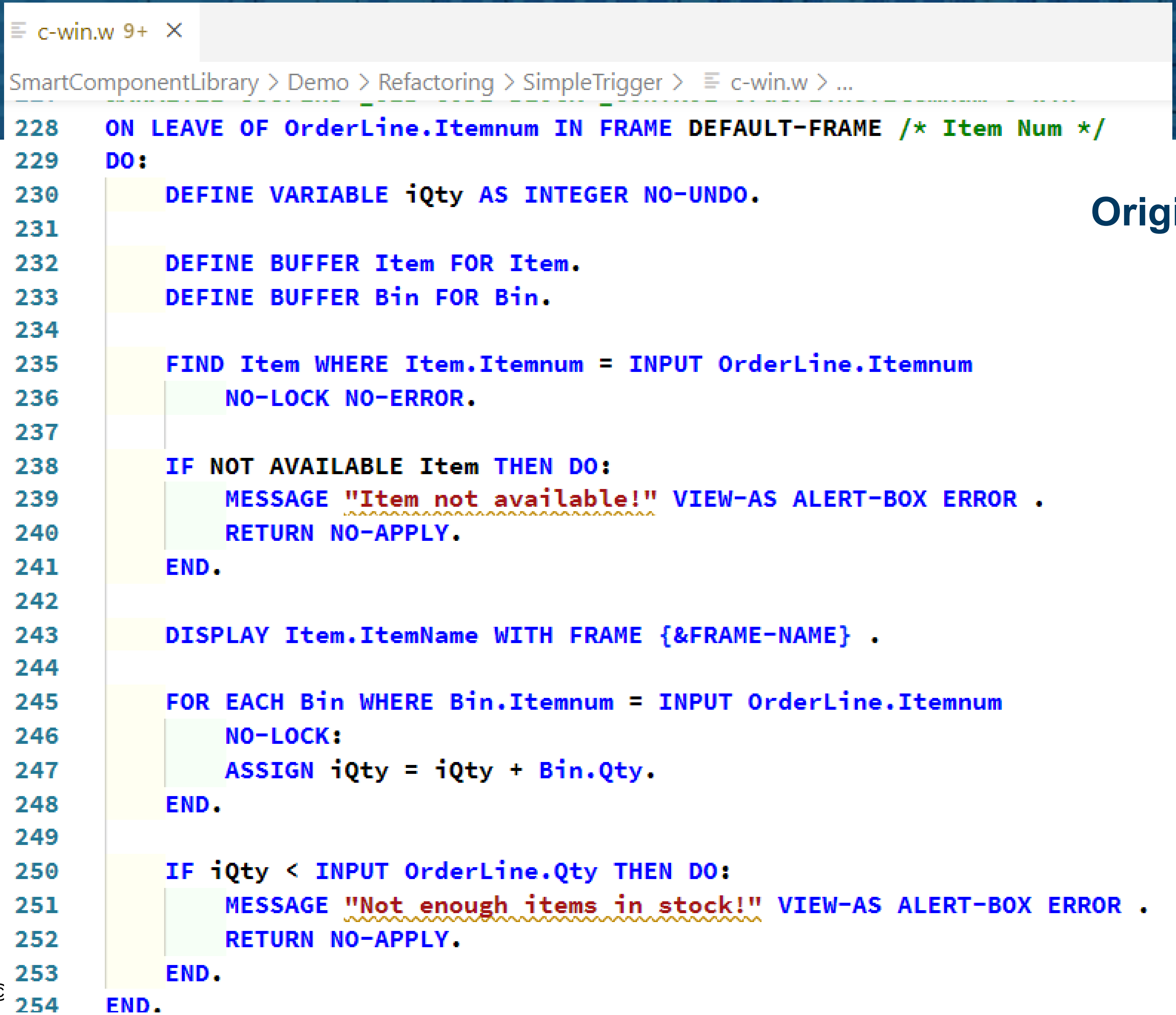


Demo

- Review different stages of the VALUE-CHANGED of OrderLine.ItemNum

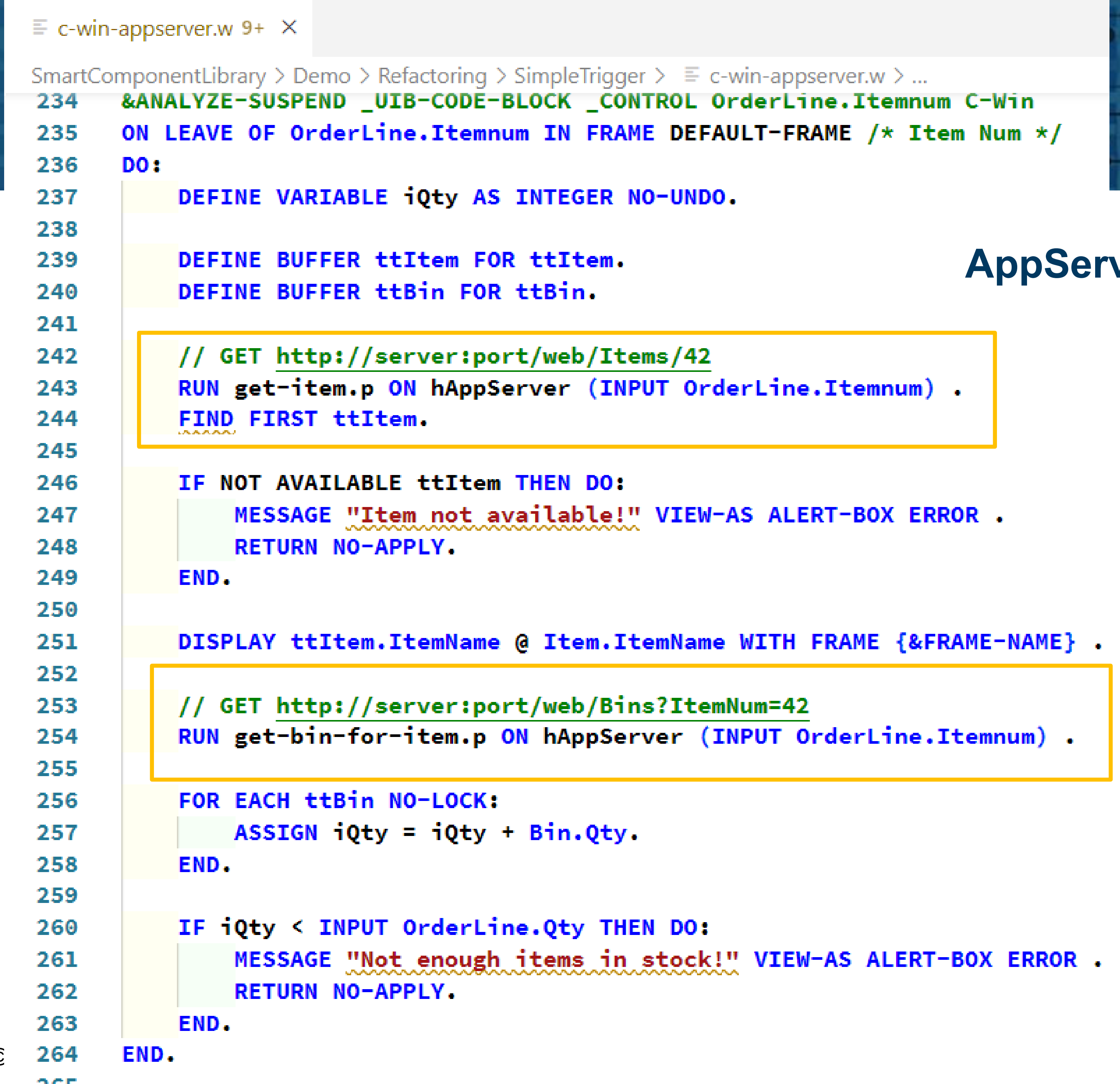
Consultingwerk
software architecture and development

Original GUI Trigger Block in .w File


c-win.w 9+
SmartComponentLibrary > Demo > Refactoring > SimpleTrigger > c-win.w > ...

```
ON LEAVE OF OrderLine.Itemnum IN FRAME DEFAULT-FRAME /* Item Num */
DO:
    DEFINE VARIABLE iQty AS INTEGER NO-UNDO.

    DEFINE BUFFER Item FOR Item.
    DEFINE BUFFER Bin FOR Bin.

    FIND Item WHERE Item.Itemnum = INPUT OrderLine.Itemnum
        NO-LOCK NO-ERROR.

    IF NOT AVAILABLE Item THEN DO:
        MESSAGE "Item not available!" VIEW-AS ALERT-BOX ERROR .
        RETURN NO-APPLY.
    END.

    DISPLAY Item.ItemName WITH FRAME {&FRAME-NAME} .

    FOR EACH Bin WHERE Bin.Itemnum = INPUT OrderLine.Itemnum
        NO-LOCK:
        ASSIGN iQty = iQty + Bin.Qty.
    END.

    IF iQty < INPUT OrderLine.Qty THEN DO:
        MESSAGE "Not enough items in stock!" VIEW-AS ALERT-BOX ERROR .
        RETURN NO-APPLY.
    END.
END.
```

AppServer Enabled Trigger Block in .w File

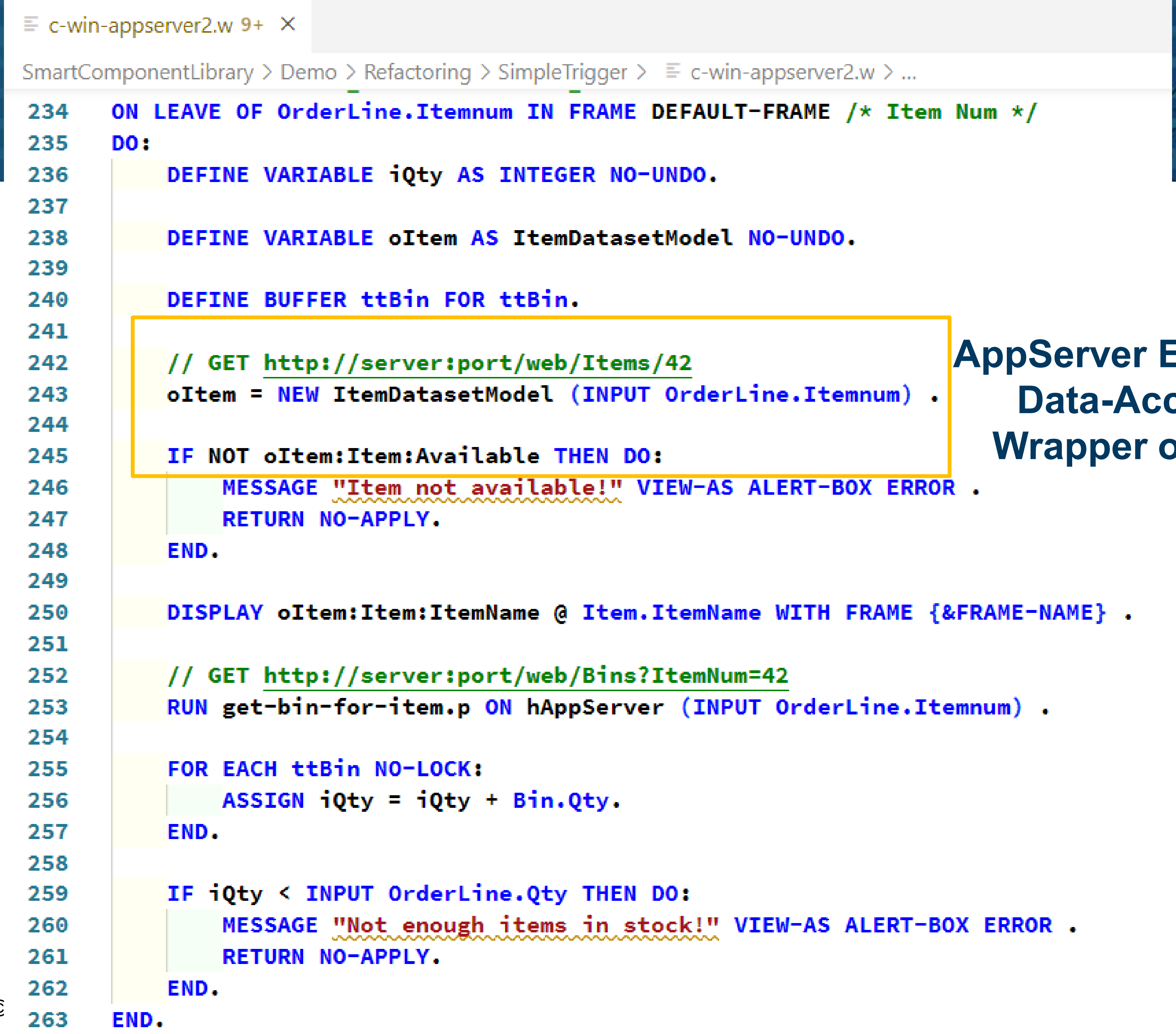


c-win-appserver.w 9+

SmartComponentLibrary > Demo > Refactoring > SimpleTrigger > c-win-appserver.w > ...

```
&ANALYZE-SUSPEND _UIB-CODE-BLOCK _CONTROL OrderLine.Itemnum C-Win
ON LEAVE OF OrderLine.Itemnum IN FRAME DEFAULT-FRAME /* Item Num */
DO:
    DEFINE VARIABLE iQty AS INTEGER NO-UNDO.

    DEFINE BUFFER ttItem FOR ttItem.
    DEFINE BUFFER ttBin FOR ttBin.

    // GET http://server:port/web/Items/42
    RUN get-item.p ON hAppServer (INPUT OrderLine.Itemnum) .
    FIND FIRST ttItem.

    IF NOT AVAILABLE ttItem THEN DO:
        MESSAGE "Item not available!" VIEW-AS ALERT-BOX ERROR .
        RETURN NO-APPLY.
    END.

    DISPLAY ttItem.ItemName @ Item.ItemName WITH FRAME {&FRAME-NAME} .

    // GET http://server:port/web/Bins?ItemNum=42
    RUN get-bin-for-item.p ON hAppServer (INPUT OrderLine.Itemnum) .

    FOR EACH ttBin NO-LOCK:
        ASSIGN iQty = iQty + Bin.Qty.
    END.

    IF iQty < INPUT OrderLine.Qty THEN DO:
        MESSAGE "Not enough items in stock!" VIEW-AS ALERT-BOX ERROR .
        RETURN NO-APPLY.
    END.
END.
```

AppServer Enabled Trigger Block in .w File
Data-Access transparent through ORM Wrapper of the SmartComponent Library

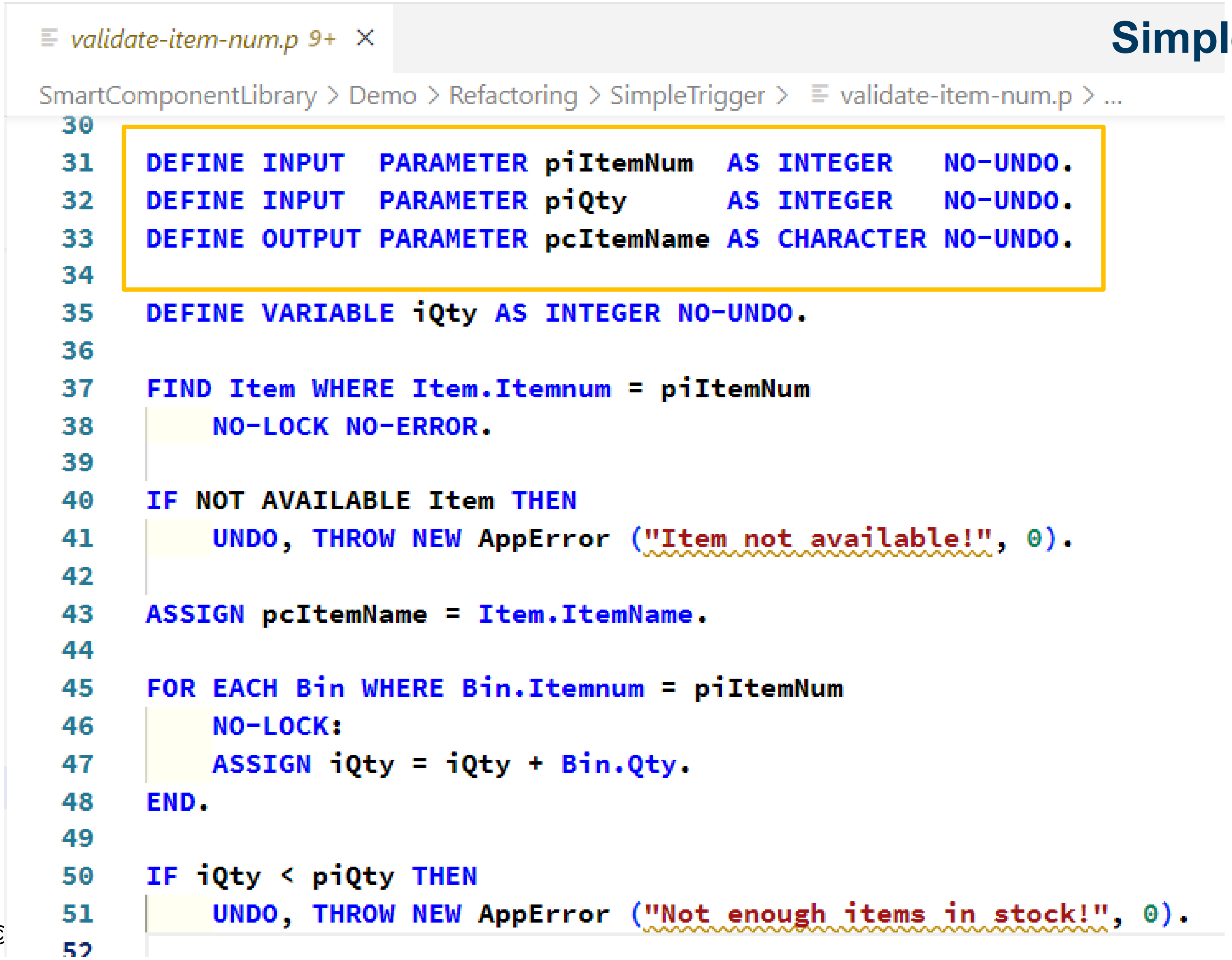


c-win-appserver2.w 9+

SmartComponentLibrary > Demo > Refactoring > SimpleTrigger > c-win-appserver2.w > ...

```
ON LEAVE OF OrderLine.Itemnum IN FRAME DEFAULT-FRAME /* Item Num */
DO:
    DEFINE VARIABLE iQty AS INTEGER NO-UNDO.

    DEFINE VARIABLE oItem AS ItemDatasetModel NO-UNDO.

    DEFINE BUFFER ttBin FOR ttBin.

    // GET http://server:port/web/Items/42
    oItem = NEW ItemDatasetModel (INPUT OrderLine.Itemnum) .

    IF NOT oItem:Item:Available THEN DO:
        MESSAGE "Item not available!" VIEW-AS ALERT-BOX ERROR .
        RETURN NO-APPLY.
    END.

    DISPLAY oItem:Item:ItemName @ Item.ItemName WITH FRAME {&FRAME-NAME} .

    // GET http://server:port/web/Bins?ItemNum=42
    RUN get-bin-for-item.p ON hAppServer (INPUT OrderLine.Itemnum) .

    FOR EACH ttBin NO-LOCK:
        ASSIGN iQty = iQty + Bin.Qty.
    END.

    IF iQty < INPUT OrderLine.Qty THEN DO:
        MESSAGE "Not enough items in stock!" VIEW-AS ALERT-BOX ERROR .
        RETURN NO-APPLY.
    END.
END.
```

Simple .p for AppServer with validation logic extracted from Trigger block


validate-item-num.p 9+ ×

SmartComponentLibrary > Demo > Refactoring > SimpleTrigger > validate-item-num.p > ...

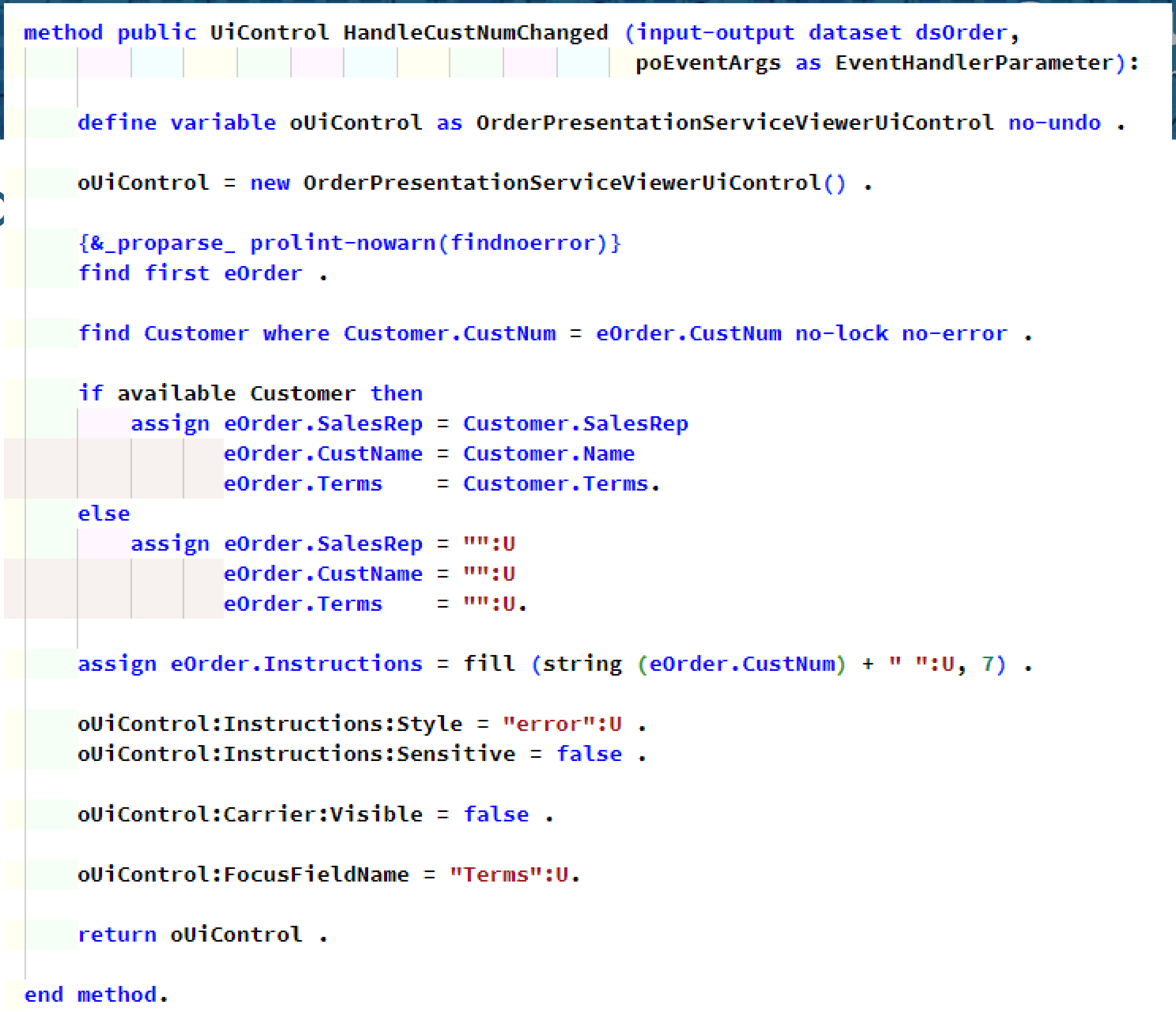
```
DEFINE INPUT  PARAMETER piItemNum  AS INTEGER   NO-UNDO.
DEFINE INPUT  PARAMETER piQty      AS INTEGER   NO-UNDO.
DEFINE OUTPUT PARAMETER pcItemName AS CHARACTER NO-UNDO.

DEFINE VARIABLE iQty AS INTEGER NO-UNDO.

FIND Item WHERE Item.Itemnum = piItemNum
    NO-LOCK NO-ERROR.

IF NOT AVAILABLE Item THEN
    UNDO, THROW NEW AppError ("Item not available!", 0).

ASSIGN pcItemName = Item.ItemName.

FOR EACH Bin WHERE Bin.Itemnum = piItemNum
    NO-LOCK:
    ASSIGN iQty = iQty + Bin.Qty.
END.

IF iQty < piQty THEN
    UNDO, THROW NEW AppError ("Not enough items in stock!", 0).
```

Consultingwerk
software architecture and development

Sample event handler – not pretty, but commonly seen

```
ON VALUE-CHANGED OF Order.CustNum IN FRAME DEFAULT-FRAME /* Cust Num */
DO:
    FIND Customer WHERE Customer.CustNum = INPUT Order.CustNum NO-LOCK NO-ERROR .

    IF AVAILABLE Customer THEN
        ASSIGN Order.SalesRep:SCREEN-VALUE = Customer.SalesRep
               Customer.Name:SCREEN-VALUE  = Customer.Name
               Order.Terms:SCREEN-VALUE    = Customer.Terms.
    ELSE
        ASSIGN Order.SalesRep:SCREEN-VALUE = "":U
               Customer.Name:SCREEN-VALUE  = "":U
               Order.Terms:SCREEN-VALUE    = "":U.

    DISPLAY FILL (STRING(Order.CustNum:INPUT-VALUE) + " ":U, 7) @ Order.Instructions WITH FRAME {&FRAME-NAME}.

    Order.Instructions:BGCOLOR = 4 .
    Order.Instructions:SENSITIVE = FALSE .

    Order.Carrier:VISIBLE = FALSE .

    APPLY "ENTRY":U TO Order.Terms IN FRAME {&FRAME-NAME}.
END.
```

Server-sic ces



```
method public UiControl HandleCustNumChanged (input-output dataset dsOrder,
                                                poEventArgs as EventHandlerParameter):

    define variable oUiControl as OrderPresentationServiceViewerUiControl no-undo .

    oUiControl = new OrderPresentationServiceViewerUiControl() .

    {&_proparse_ prolint-nowarn(findnoerror)}
    find first eOrder .

    find Customer where Customer.CustNum = eOrder.CustNum no-lock no-error .

    if available Customer then
        assign eOrder.SalesRep = Customer.SalesRep
               eOrder.CustName = Customer.Name
               eOrder.Terms    = Customer.Terms.
    else
        assign eOrder.SalesRep = "":U
               eOrder.CustName = "":U
               eOrder.Terms    = "":U.

    assign eOrder.Instructions = fill (string (eOrder.CustNum) + " ":U, 7) .

    oUiControl:Instructions:Style = "error":U .
    oUiControl:Instructions:Sensitive = false .

    oUiControl:Carrier:Visible = false .

    oUiControl:FocusFieldName = "Terms":U.

    return oUiControl .

end method.
```

Unit Testing of event-handler

- “Event-handler” now much simpler for unit-test
- No dependency on actual user-interface
- No direct dependency on database allows “mocking” of data or data access

Consultingwerk
software architecture and development

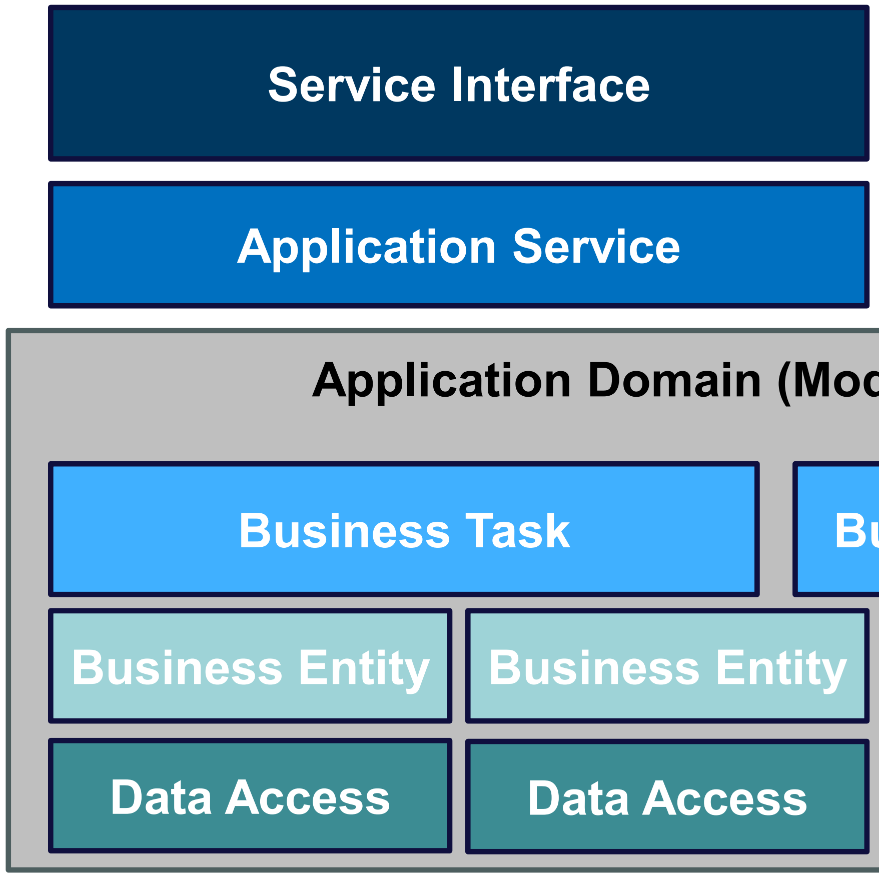
```
{Consultingwerk/SmartComponentsDemo/OERA/Sports2000/dsOrder.i}

@Test.
method public void TestMethod():

    define variable oService    as OrderPresentationService no-undo.
    define variable oUiControl as UiControl                no-undo.

    oService = new OrderPresentationService().

    dataset dsOrder:read-xml ("file":u,
                              "Consultingwerk/SmartComponentsDemo/PresentationService/test.xml":u,
                              ?, ?, ?).

    find first eOrder.
    assign eOrder.CustNum = 1 . /* new screen-value */

    oUiControl = oService:HandleCustNumChanged (dataset dsOrder,
                                                new EventHandlerParameter ("CustNum":u, "ValueChanged":u)).

    find first eOrder.

    Assert:Equals(eOrder.CustName, "Lift Line Skiing":u) .
    Assert:Equals(oUiControl:FieldControl("Instructions":u):Style, "error":u) .

end method.
```

Benefits of top-down code-generalization

- First introduce service-ready component based on existing business logic
- Hide implementation details behind service interface
- Flow of business logic remains largely the same – **this will reduce risk**
- Component interface will allow
  - Use in modern user-interfaces
  - Implementation of unit-tests
- Unit tests will improve confidence when optimizing the code

Aspects of Top-Down code generalization

- Business Tasks and Business Entities should only deal with “their concern”
- Use factories or service managers – never directly new any application or domain business service object
- Only “allow” calls from top to bottom
- Services within a domain may call each other
- Services across domain boundary should use domain service interface



Further considerations

- Use parameter objects
- Separate session or screen-context from request parameters
- Selected warehouse may be session context
- Selected warehouse may be screen context (might be a screen setting)
- Screen-context might be modified in UI and backend
- Selected order may be request-context (it’s the subject of ship order)

- Variables defined in the “definitions section” vs. parameters to internal procedures

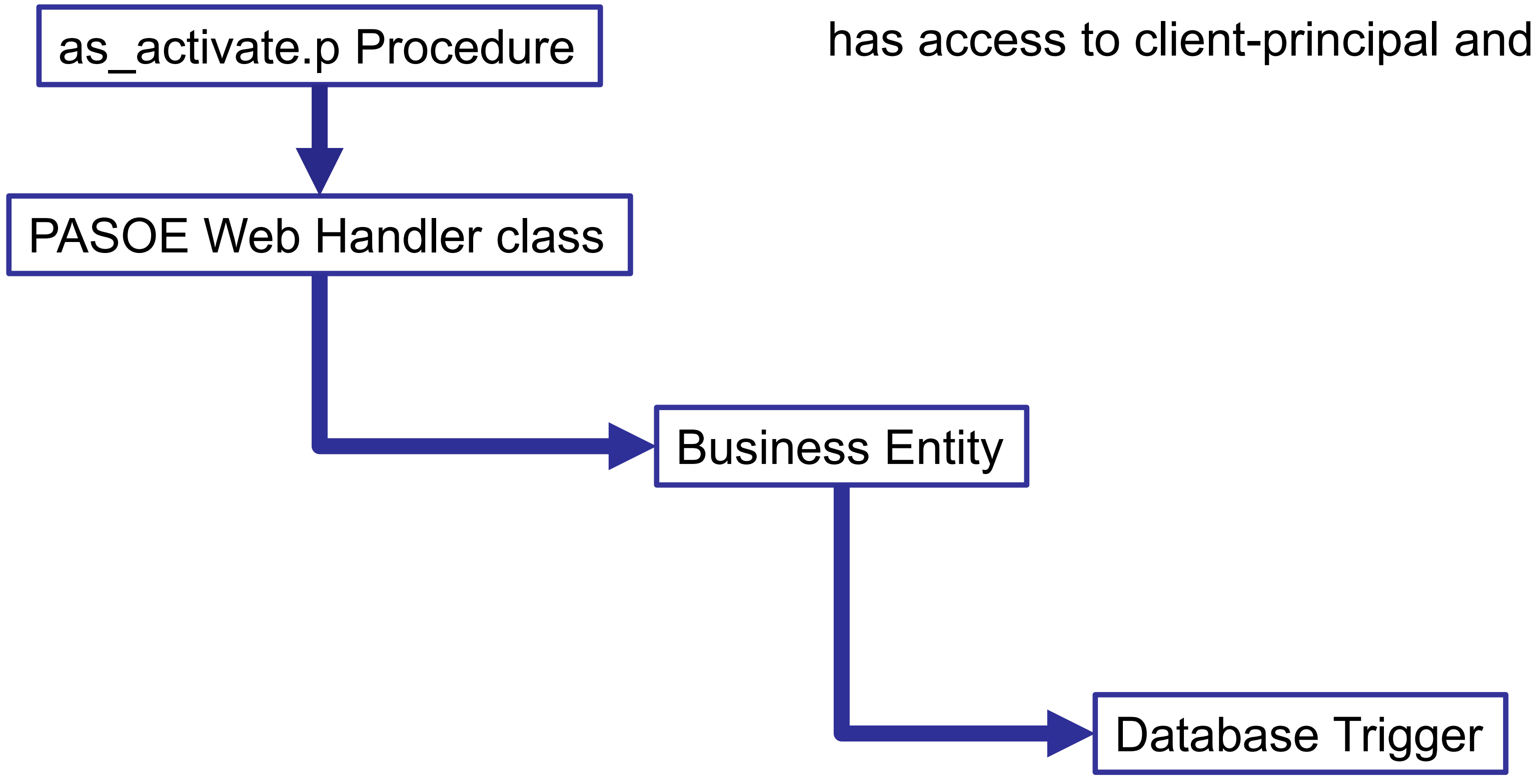
Agenda

- Modernization Process
- Application Architecture
- **Dealing with (GLOBAL) SHARED Variables**
- Dealing with messages or prompts
- Proparse
- Record Locking

GLOBAL SHARED or SHARED variables …

- GLOBAL SHARED variables are less trouble
- SHARED variables should be reconsidered – many of them may be replaced with GLOBAL SHARED, usually a bad legacy
- Class based code (most new code, PASOE Web handlers) has NO access to any GLOBAL SHARED SHARED context

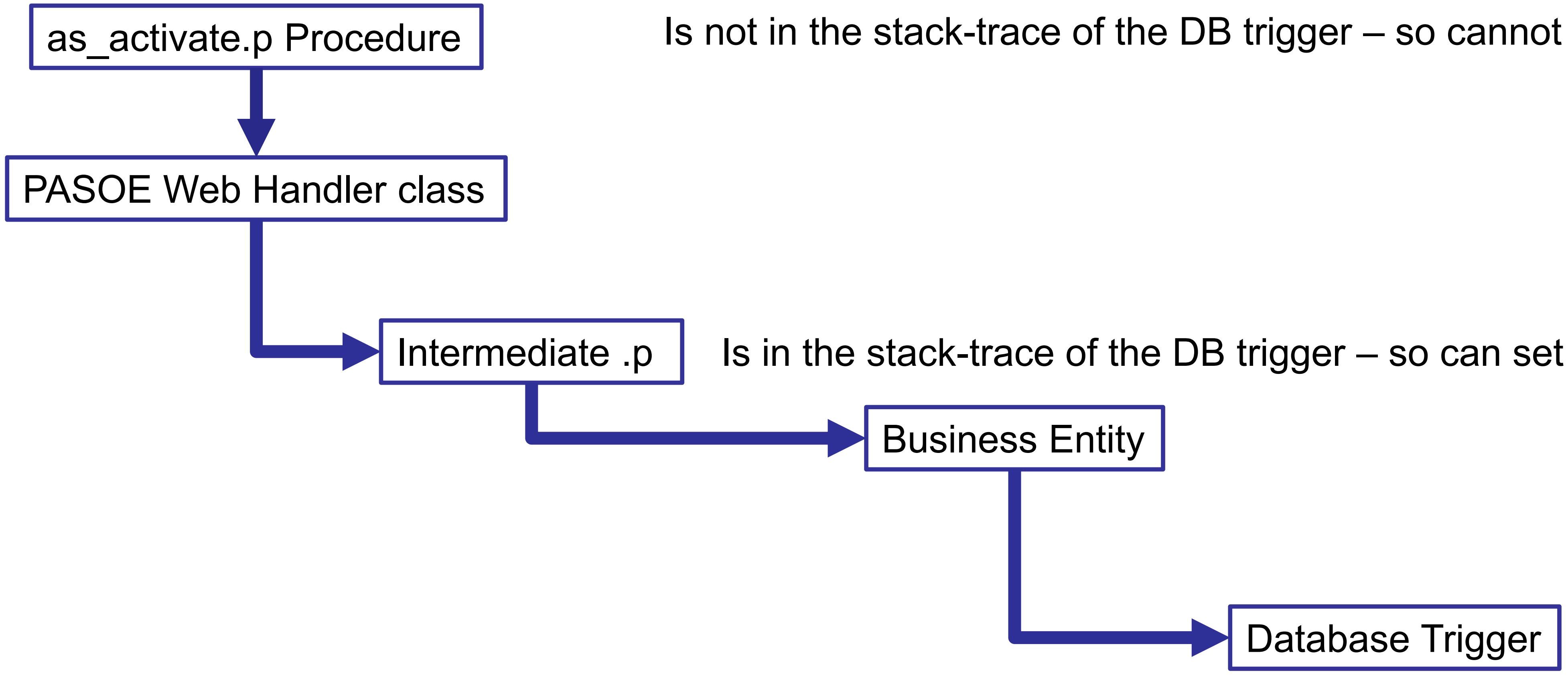
DB Trigger relying on a GLOBAL SHARED variable



has access to client-principal and global-shared

can use global-shared

DB Trigger relying on a SHARED variable



Is not in the stack-trace of the DB trigger – so cannot set SHARED var

Is in the stack-trace of the DB trigger – so can set NEW SHARED var

can use shared

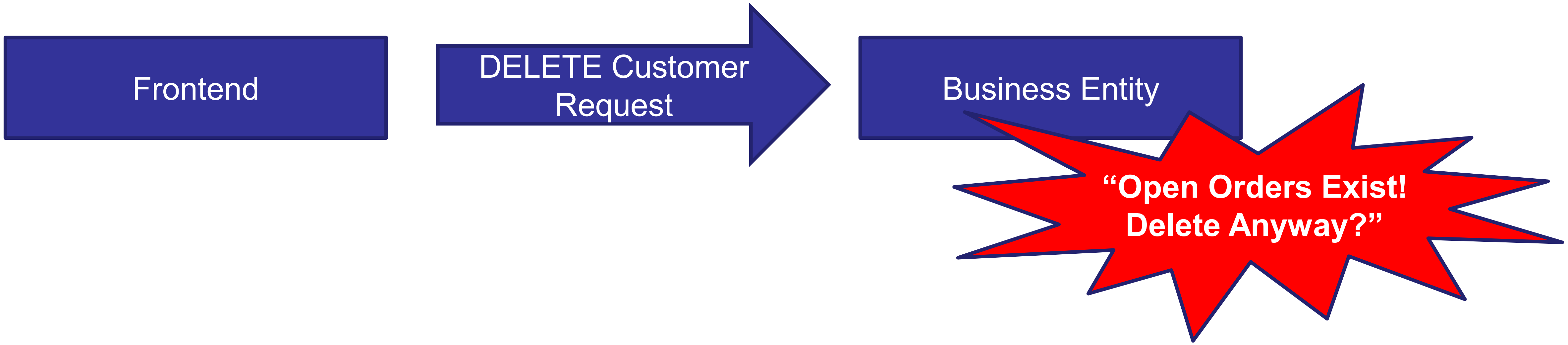
Agenda

- Modernization Process
- Application Architecture
- Dealing with (GLOBAL) SHARED Variables
- **Dealing with messages or prompts**
- Proparse
- Record Locking

Input blocking from the Backend

- Progress Application Server does not support Input Blocking on the UI
- Once AppServer is invoked, client waits for response
- Web technologies such as Socket.IO may be used to send messages from Backend to frontend
  - Back not vice-versa, no WAIT-FOR
- **When UI can foresee** that AppServer **may** require additional information when processing request, try adding this to the request
  - However UX should not be ignored. Too many irrelevant options confusion / annoying to users

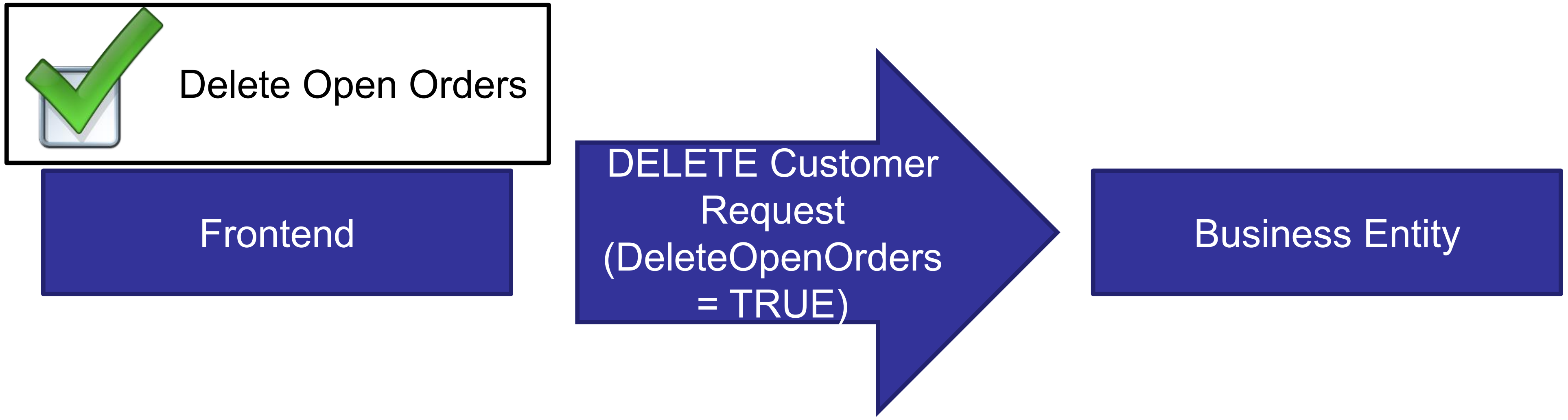
Input Blocking, fat client ABL



```
IF CAN-FIND (FIRST Order OF Customer WHERE ....) THEN
    MESSAGE "Open Orders Exist! Delete Anyway?"
        VIEW-AS ALERT-BOX QUESTION BUTTONS YES-NO UPDATE response .
```

Consultingwerk
software architecture and development

Input Blocking, fat client ABL



```
IF CAN-FIND (FIRST Order OF Customer WHERE ....)
   AND poRequest:DeleteOpenOrders = TRUE THEN …
```

Example challenge: Interaction between Back and Frontend

- Assumption: Existing Business Logic in large parts suitable as foundation for new application (functional and structural), especially validation
- Validation may also provide color coding to represent field status etc.
- Validation may have to prompt the user
- Web applications typically:
  Request (from browser) – Response (from server)
- No Input-Blocking (not possible to wait for user input in Business Logic)

Sample: Yes/No PROMPT in validation

- Demand is to keep the validation flow in major parts “as is“
- Validation may encounter question requiring user input: “Are you sure?” etc.

Sample: Yes/No PROMPT in validation

```
/* ---------- */
/* Verstorben */
/* ---------- */
if (date(Stamm.Todes_Dat:screen-value) <> ?) then do:
  /* Testen, ob Versicherter gerade eben verstorben ist. */
  if (EDIT_MODE = "UPDATE") then do:
    find Stamm no-lock where recid(Stamm) = MAIN_REC_ID.
    if (Stamm.Todes_Dat = ?) then do:
      /* Versicherter wurde soeben auf verstorben gesetzt. */
      run set_message_param(Stamm.Todes_Dat:screen-value).
      run user_warning("Der Versicherte ist am $1 verstorben. ~n~n" +
                       "Die zugehörigen Wohnadressen werden gesperrt.~n" +
                       "Überprüfen Sie, ob noch Revisionen vorgesehen sind~n" +
                       "und/oder Hilfsmittel zurückgenommen werden müssen.~n",
                       output continue).
      if not continue then return error.
    end.
  end.

end. /* if verstorben */
```

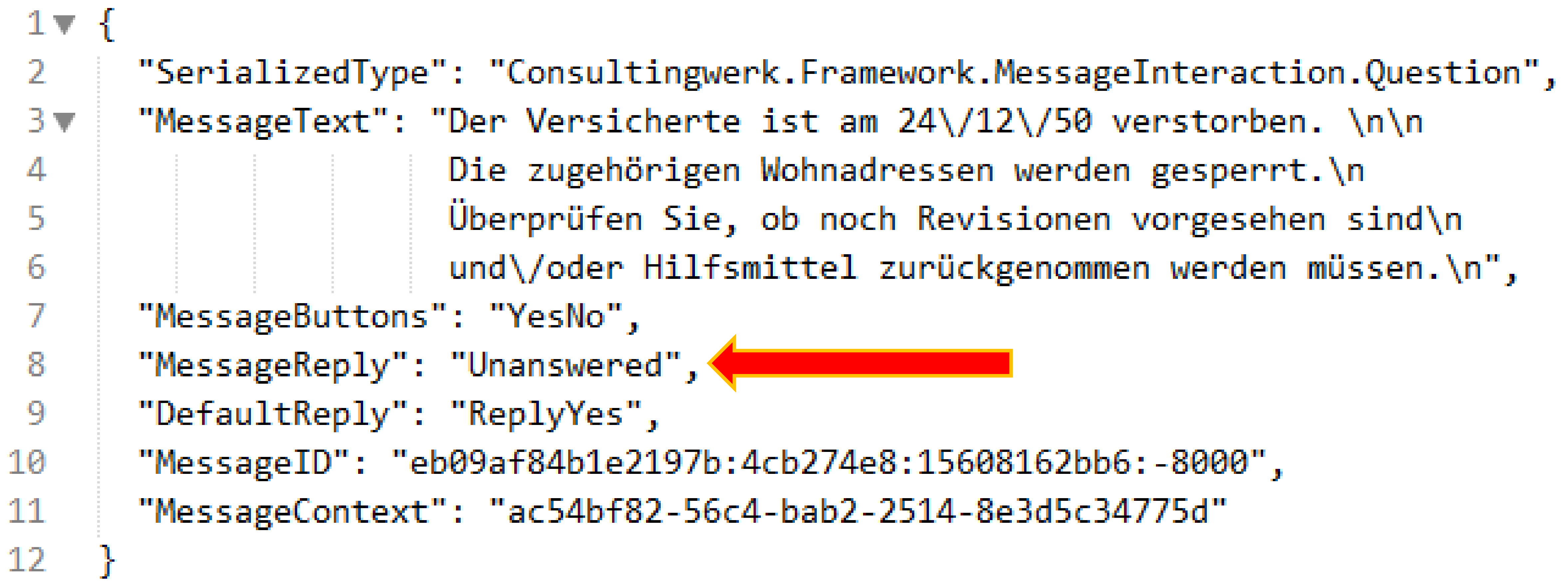
Sample: Yes/No PROMPT in validation

```
MSG = {Consultingwerk/get-service.i IMsg} .
SYS = {Consultingwerk/get-service.i ISys} .
MOD_ADD = {Consultingwerk/get-service.i IModAdd} .

if (eStammBefore.Todes_Dat = ?) then do:
    /* Versicherter wurde soeben auf verstorben gesetzt. */
    MSG:set_message_param(string (eStamm.Todes_Dat) /*:screen-value*/).

    continue = MSG:user_warning("Der Versicherte ist am $1 verstorben. ~n~n" +
                                "Die zugehörigen Wohnadressen werden gesperrt.~n" +
                                "Überprüfen Sie, ob noch Revisionen vorgesehen sind~n" +
                                "und/oder Hilfsmittel zurückgenommen werden müssen.~n",
                                this-object:GetClass():TypeName,
                                "eb09af84b1e2197b:4cb274e8:15608162bb6:-8000",
                                string (eStamm.SelfHdl)).

    if not continue then do:
        DatasetHelper:AddErrorString(buffer eStamm:handle, "_CANCEL") .
        return .
    end.

    /*if not continue then return error.*/
end.
```

Migration using MessageInteractionService API (SmartComponent Library framework)

- Backend – API maintains list of questions (unanswered and answered)
- Same API Call may ask a new question or return an existing answer
- Supports multiple questions per routine: Questions are flagged with e.g. a GUID identifying their location in code
- Support for multiple iterations (Loops, FOR EACH, …): Each question is also flagged with a records PUK value (GUID, combined key fields)

JSON Representation of the question



```
{
  "SerializedType": "Consultingwerk.Framework.MessageInteraction.Question",
  "MessageText": "Der Versicherte ist am 24\/12\/50 verstorben. \n\n
                  Die zugehörigen Wohnadressen werden gesperrt.\n
                  Überprüfen Sie, ob noch Revisionen vorgesehen sind\n
                  und\/oder Hilfsmittel zurückgenommen werden müssen.\n",
  "MessageButtons": "YesNo",
  "MessageReply": "Unanswered",
  "DefaultReply": "ReplyYes",
  "MessageID": "eb09af84b1e2197b:4cb274e8:15608162bb6:-8000",
  "MessageContext": "ac54bf82-56c4-bab2-2514-8e3d5c34775d"
}
```

Automation

- Migration of MESSAGE Statements into API calls can be automated using Proparse based tooling

Agenda

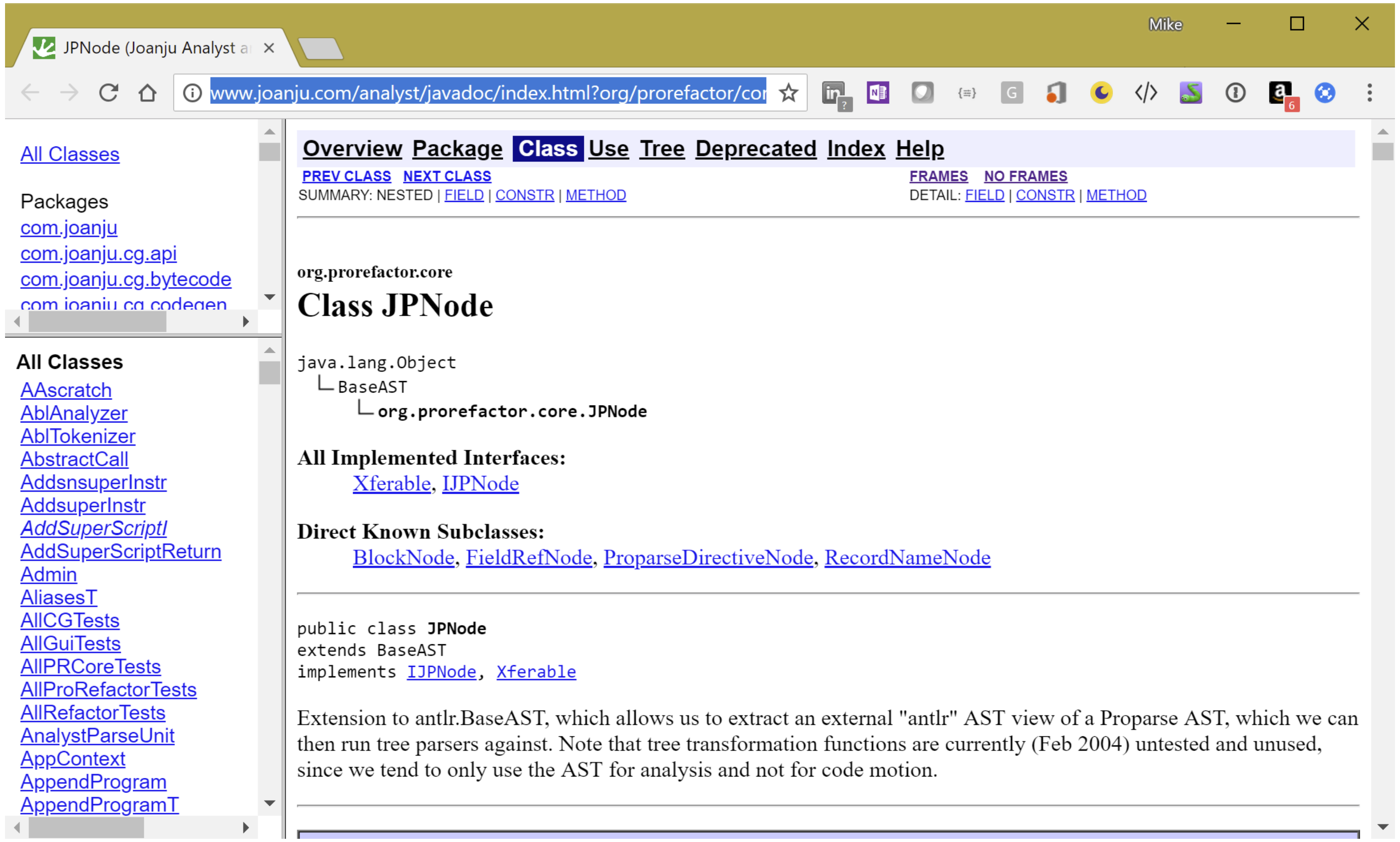
- Modernization Process
- Application Architecture
- Dealing with (GLOBAL) SHARED Variables
- Dealing with messages or prompts
- **Proparse**
- Record Locking

Source code parsing using Proparse

- ABL syntax parser, abstract view on ABL source code, based on ANTLR
- Eliminates the need for text based source code analysis
  - Resolves issues with line-breaks, abbreviated keywords, mixed order of keywords
- Open source
  - github.com/oehive/proparse
  - github.com/consultingwerk/proparse
  - github.com/riverside-software/proparse
- Actively maintained in various forks, support for 12.8 ABL syntax

Proparse

- http://www.joanju.com/analyst/javadoc/index.html?org/prorefactor/core/JPNode.html



JPNode (Joanju Analyst a

www.joanju.com/analyst/javadoc/index.html?org/prorefactor/cor

All Classes

Packages
com.joanju
com.joanju.cg.api
com.joanju.cg.bytecode
com.joanju.cg.codegen

All Classes
AAscratch
AblAnalyzer
AblTokenizer
AbstractCall
AddsnsuperInstr
AddsuperInstr
AddSuperScriptI
AddSuperScriptReturn
Admin
AliasesT
AllCGTests
AllGuiTests
AllPRCoreTests
AllProRefactorTests
AllRefactorTests
AnalystParseUnit
AppContext
AppendProgram
AppendProgramT

Overview Package **Class** Use Tree Deprecated Index Help
PREV CLASS NEXT CLASS FRAMES NO FRAMES
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

org.prorefactor.core
Class JPNode

```
java.lang.Object
  └BaseAST
      └org.prorefactor.core.JPNode
```

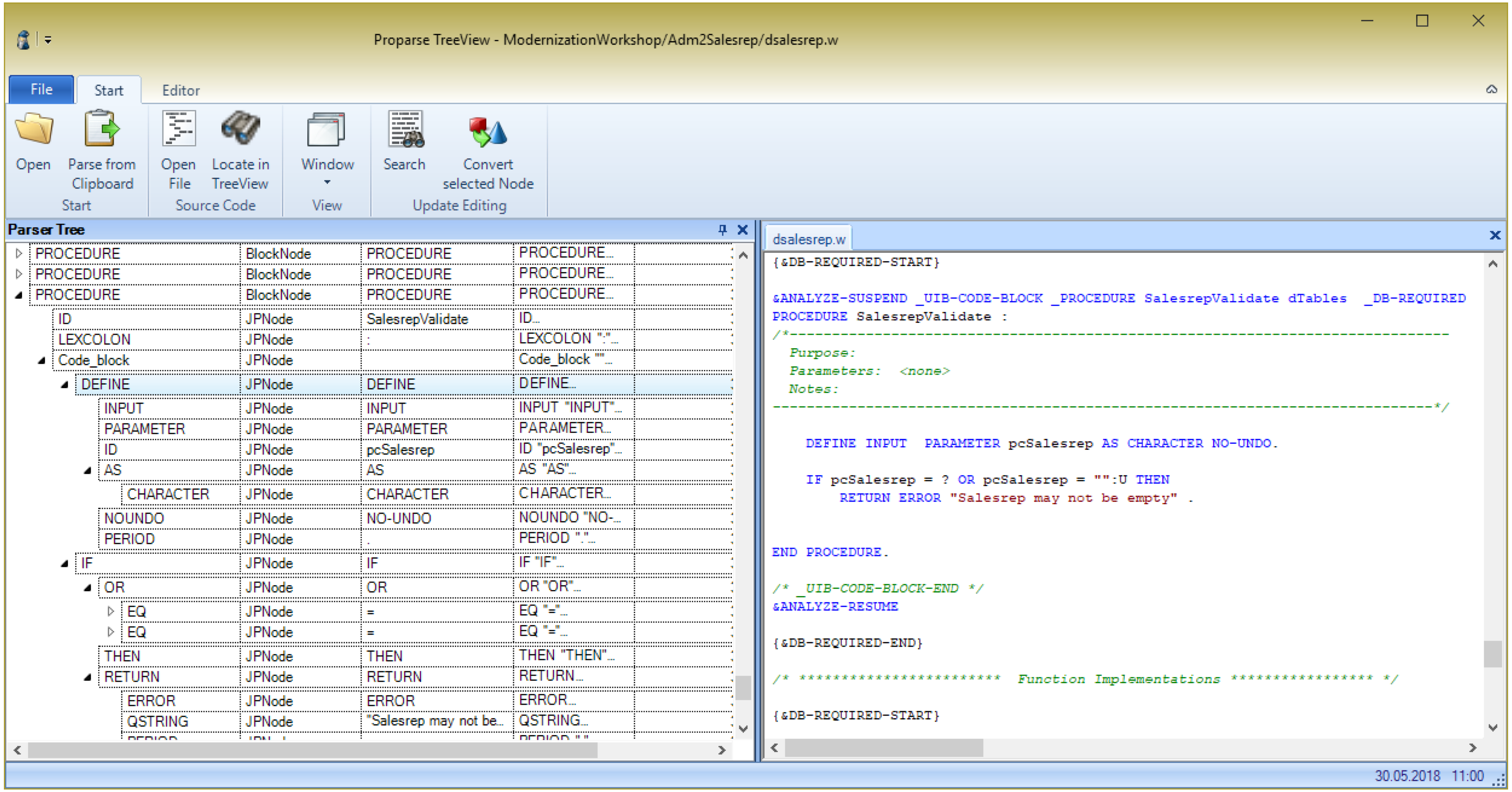
All Implemented Interfaces:
Xferable, IJPNode

Direct Known Subclasses:
BlockNode, FieldRefNode, ProparseDirectiveNode, RecordNameNode

```
public class JPNode
extends BaseAST
implements IJPNode, Xferable
```

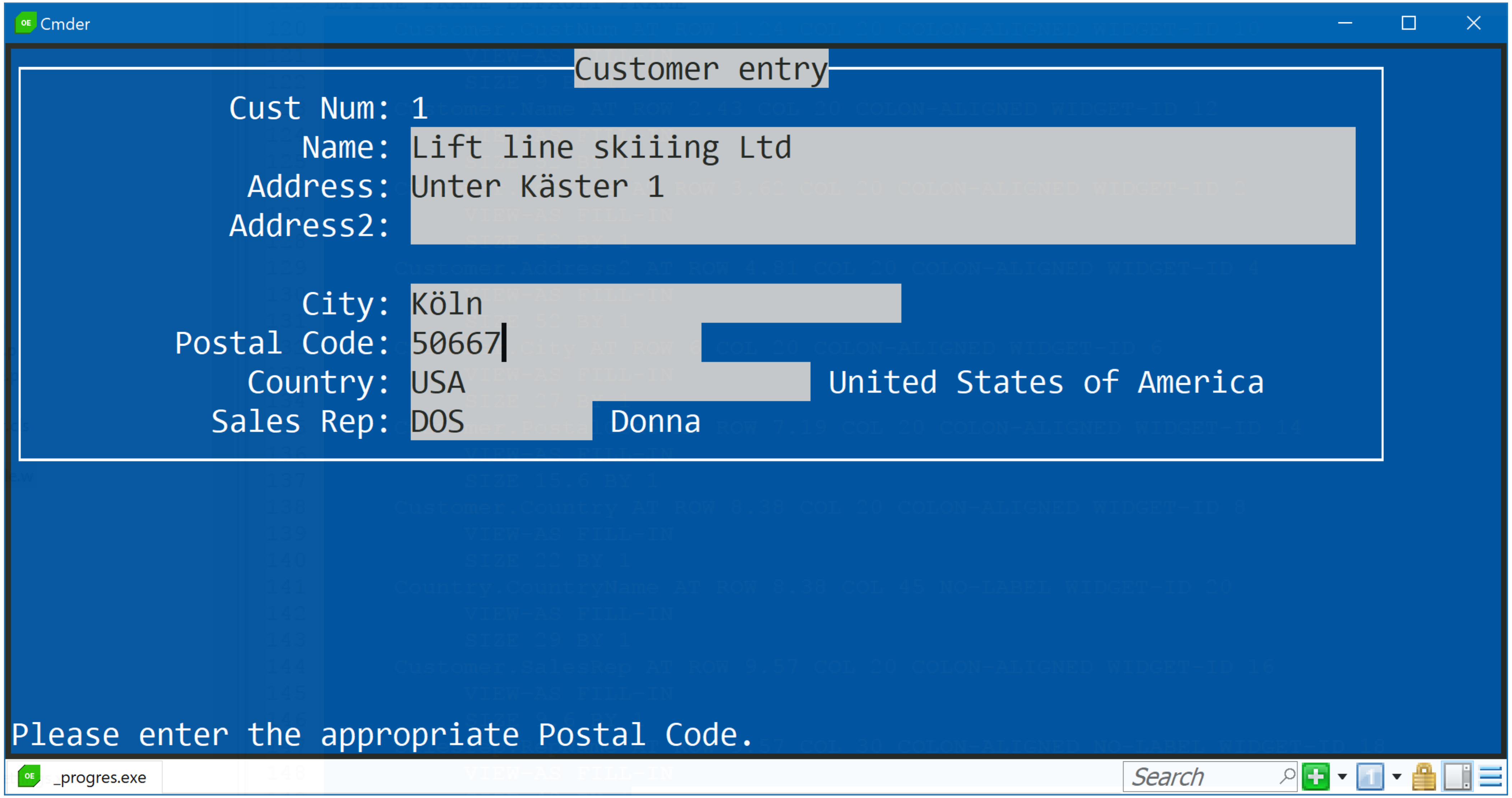
Extension to antlr.BaseAST, which allows us to extract an external "antlr" AST view of a Proparse AST, which we can then run tree parsers against. Note that tree transformation functions are currently (Feb 2004) untested and unused, since we tend to only use the AST for analysis and not for code motion.

```
javafile = NEW java.io.File (pcFilename).

IF (NOT javafile:exists()) THEN
    UNDO, THROW NEW FileNotFoundException (pcFileName,
                                           SUBSTITUTE ("Could not find file: &1."{&TRAN}, pcFileName),
                                           0) .

IF cProparseCodepage > "" THEN DO:
    IF NOT Codepages:IsKnownCodepage (cProparseCodepage) THEN
        UNDO, THROW NEW InvalidValueException (cProparseCodepage, "ProparseCodepage":U) .

    oParseUnit = NEW ParseUnit(javafile, cProparseCodepage).
END.
ELSE
    oParseUnit = NEW ParseUnit(javafile).

pu:treeParser01().

DELETE OBJECT javafile .
```



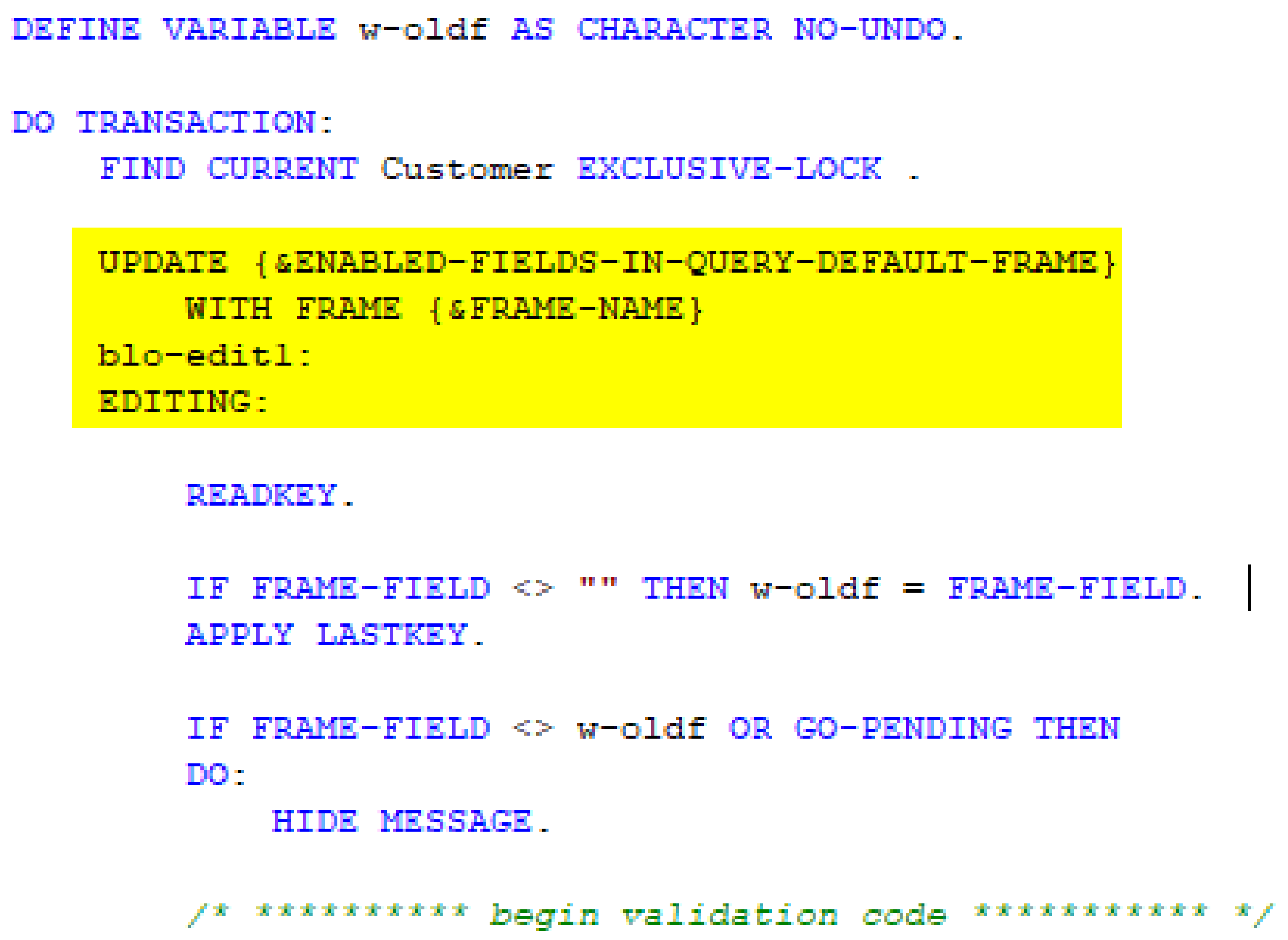
PROCEDURE	BlockNode	PROCEDURE	PROCEDURE...
PROCEDURE	BlockNode	PROCEDURE	PROCEDURE...
PROCEDURE	BlockNode	PROCEDURE	PROCEDURE...
ID	JPNode	SalesrepValidate	ID...
LEXCOLON	JPNode	:	LEXCOLON ":"...
Code_block	JPNode		Code_block ""...
DEFINE	JPNode	DEFINE	DEFINE...
INPUT	JPNode	INPUT	INPUT "INPUT"...
PARAMETER	JPNode	PARAMETER	PARAMETER...
ID	JPNode	pcSalesrep	ID "pcSalesrep"...
AS	JPNode	AS	AS "AS"...
CHARACTER	JPNode	CHARACTER	CHARACTER...
NOUNDO	JPNode	NO-UNDO	NOUNDO "NO-...
PERIOD	JPNode	.	PERIOD "."...
IF	JPNode	IF	IF "IF"...
OR	JPNode	OR	OR "OR"...
EQ	JPNode	=	EQ "="...
EQ	JPNode	=	EQ "="...
THEN	JPNode	THEN	THEN "THEN"...
RETURN	JPNode	RETURN	RETURN...
ERROR	JPNode	ERROR	ERROR...
QSTRING	JPNode	"Salesrep may not be...	QSTRING...

```
{&DB-REQUIRED-START}

&ANALYZE-SUSPEND _UIB-CODE-BLOCK _PROCEDURE SalesrepValidate dTables  _DB-REQUIRED
PROCEDURE SalesrepValidate :
/*------------------------------------------------------------------------------
  Purpose:
  Parameters:  <none>
  Notes:
------------------------------------------------------------------------------*/

    DEFINE INPUT  PARAMETER pcSalesrep AS CHARACTER NO-UNDO.

    IF pcSalesrep = ? OR pcSalesrep = "":U THEN
        RETURN ERROR "Salesrep may not be empty" .


END PROCEDURE.

/* _UIB-CODE-BLOCK-END */
&ANALYZE-RESUME

{&DB-REQUIRED-END}

/* ************************  Function Implementations ***************** */

{&DB-REQUIRED-START}
```



```
                 Customer entry
   Cust Num: 1
       Name: Lift line skiiing Ltd
    Address: Unter Käster 1
   Address2:

       City: Köln
Postal Code: 50667
    Country: USA           United States of America
  Sales Rep: DOS     Donna

Please enter the appropriate Postal Code.
```

UPDATE EDITING Blocks



```
DEFINE VARIABLE w-oldf AS CHARACTER NO-UNDO.

DO TRANSACTION:
    FIND CURRENT Customer EXCLUSIVE-LOCK .

    UPDATE {&ENABLED-FIELDS-IN-QUERY-DEFAULT-FRAME}
        WITH FRAME {&FRAME-NAME}
    blo-editl:
    EDITING:

        READKEY.

        IF FRAME-FIELD <> "" THEN w-oldf = FRAME-FIELD.
        APPLY LASTKEY.

        IF FRAME-FIELD <> w-oldf OR GO-PENDING THEN
        DO:
            HIDE MESSAGE.

        /* *********** begin validation code ************ */
```

Single field validation within EDITING Block

```
IF w-oldf = "Salesrep" OR GO-PENDING THEN DO:

    FIND Salesrep WHERE Salesrep.SalesRep = INPUT Customer.SalesRep
        NO-LOCK NO-ERROR .

    IF NOT AVAILABLE Salesrep THEN DO:
        MESSAGE SUBSTITUTE ("Please enter a valid salesrep code. &1 is not a valid salesrep code.",
                            INPUT Customer.Salesrep) .
        NEXT-PROMPT Customer.Salesrep WITH FRAME {&frame-name}.
        NEXT blo-edit1.
    END.
    ELSE
        DISPLAY UPPER (Salesrep.SalesRep) @ Customer.SalesRep
                Salesrep.RepName WITH FRAME {&frame-name} .
END.
```

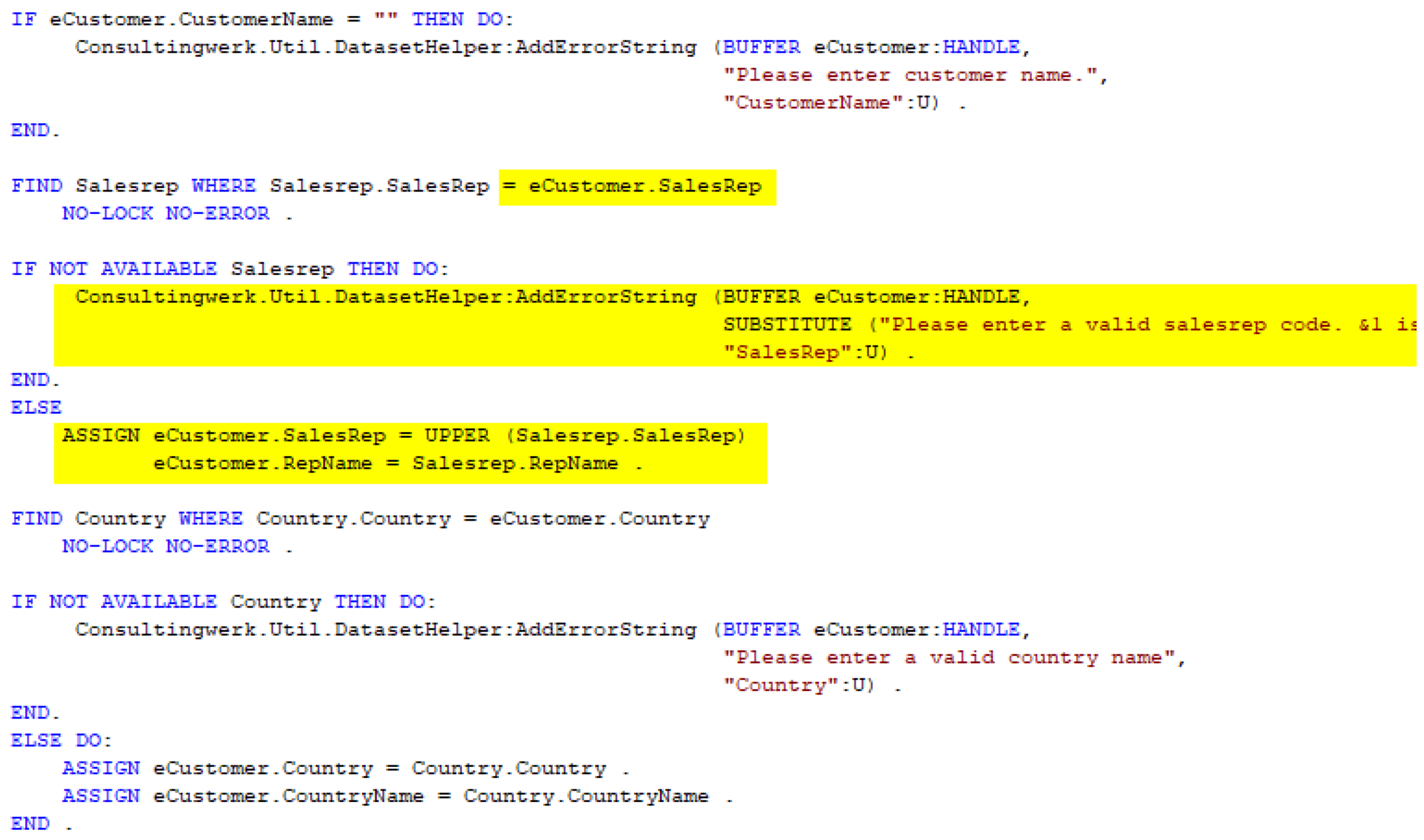
UPDATE EDITING Blocks

- Commonly used in TTY and early GUI applications
- Full of validation logic / Lookup functionality (locating foreign key descriptions)
- Tied to UI through “INPUT <fieldname>” references
- MESSAGE Statement used for error messages
- NEXT-PROMPT provides field that should receive input after error
- Record locked during duration of the UPDATE Statement

UPDATE EDITING Blocks

- Iterated for every keystroke or GO-PENDING
- When invoked on GO-PENDING, it’s similar to a commit to a Business Entity
  - Validating all fields at once
  - Processing update when no validation error occurred
  - Returning validation error to user (with instruction of next field)
- Code flow in EDITING Block very similar to typical Business Entity validation

Business Entity Validation based on UPD EDITING



```
IF eCustomer.CustomerName = "" THEN DO:
     Consultingwerk.Util.DatasetHelper:AddErrorString (BUFFER eCustomer:HANDLE,
                                                       "Please enter customer name.",
                                                       "CustomerName":U) .
END.

FIND Salesrep WHERE Salesrep.SalesRep = eCustomer.SalesRep
    NO-LOCK NO-ERROR .

IF NOT AVAILABLE Salesrep THEN DO:
     Consultingwerk.Util.DatasetHelper:AddErrorString (BUFFER eCustomer:HANDLE,
                                                       SUBSTITUTE ("Please enter a valid salesrep code. &1 is
                                                       "SalesRep":U) .
END.
ELSE
    ASSIGN eCustomer.SalesRep = UPPER (Salesrep.SalesRep)
           eCustomer.RepName = Salesrep.RepName .

FIND Country WHERE Country.Country = eCustomer.Country
    NO-LOCK NO-ERROR .

IF NOT AVAILABLE Country THEN DO:
     Consultingwerk.Util.DatasetHelper:AddErrorString (BUFFER eCustomer:HANDLE,
                                                       "Please enter a valid country name",
                                                       "Country":U) .
END.
ELSE DO:
    ASSIGN eCustomer.Country = Country.Country .
    ASSIGN eCustomer.CountryName = Country.CountryName .
END .
```

Business Entity Validation based on UPD EDITING

- IF w-oldf OR GO-ENDING not required; Business Entity typically validates all fields at once
  - Removing at least one level of blocks in the code
- *“INPUT <fieldname>”* replaced with temp-table field reference
- *DISPLAY* statements replaces with update of temp-table field
- *MESSAGE/NEXT-PROMPT* statements replaced with API call to return validation message to the consumer of the Business Entity and control target field

Demo

- Proparse based migration of UPDATE EDITING Blocks into Business Entity Validation block

Agenda

- Modernization Process
- Application Architecture
- Dealing with (GLOBAL) SHARED Variables
- Dealing with messages or prompts
- Proparse
- **Record Locking**

Record Locking

- Record locking and Transaction concepts in the ABL within AppServer requests working as usual
- Legacy applications traditionally using pessimistic locking
- In an ABL fat-client with AppServer support scenario, ABL client and AppServer can lock records from each other …
- Different AppServer sessions serving the same client could lock records from each other …
- Minimum –lkwtmo of 10 seconds not ideal for AppServer requests

Record locking

- AppServer requests work better with optimistic locking – avoid record locking for long between AppServer request, detects update collisions when trying to update record (ProDataset before-image, time-stamp, etc.)

- Functional requirements may include record locking in distributed applications, e.g., ensure that Order header is not modified while updating or processing Order lines or related data
  - May be required to ensure record integrity
  - Item prices dependent on Terms in Order header

Soft record locks

- Alternative to record locks and transactions _can_ be soft locks
- Database table (e.g. SmartLock) with
  - Session Identifier
  - User Identifier
  - Resource Identifier
    - Database Table name and PUK values, “sports2000.Order” and “42”
    - Logical resource name: “month end processing 09/2024”
  - Lock time-out
- Time-out used to avoid eternal locks, alternative to back out locks on client disconnect

Soft lock API

- Acquire lock: Obtain record lock
  - Verify no other session is holding a lock record for the resource
  - Create lock record
  - Update existing lock record to refresh time-out
  - Return TRUE/FALSE or throw error
- Release lock:
  - Verify this session is holding the lock
  - Delete lock record
- Release all session locks:
  - Delete all lock records of a session on disconnect

Soft Lock Support

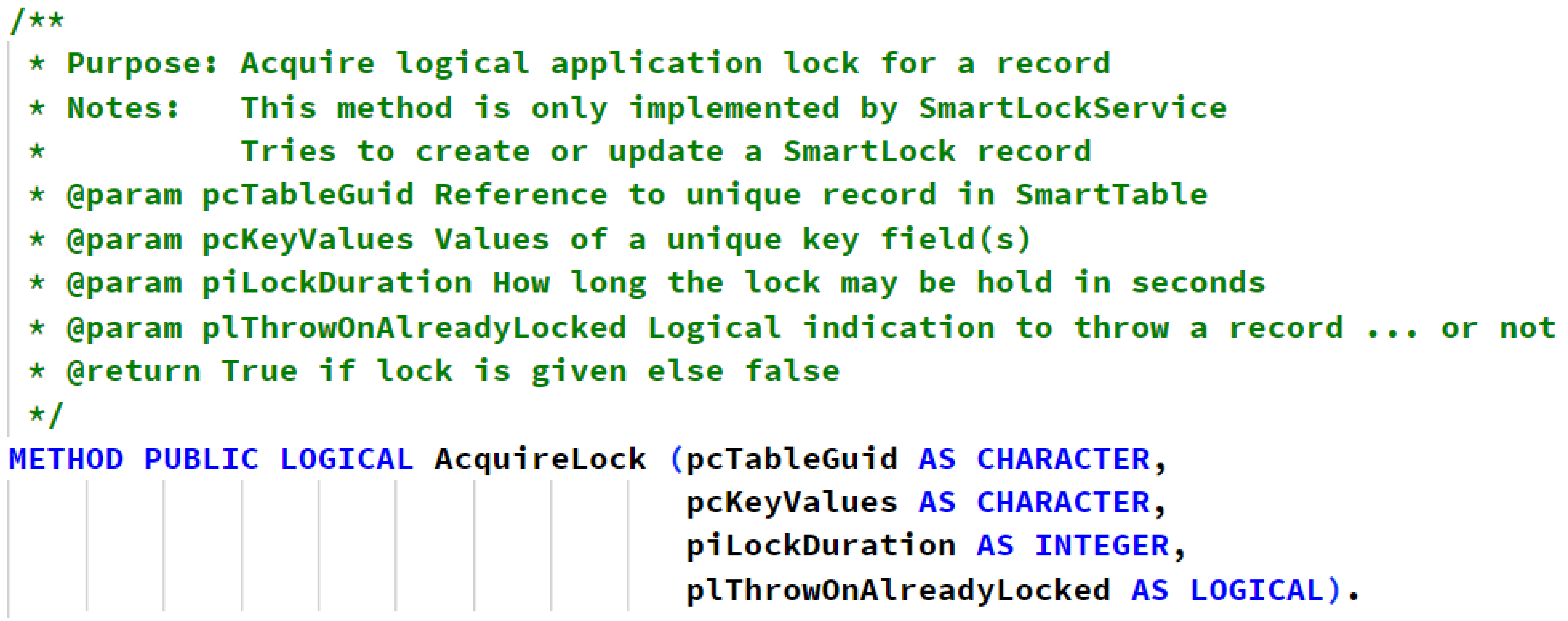
- Implement a scheduler job (e.g. all 15 minutes) to wipe out all expired lock records
- Consider implementing soft lock API also in legacy application as required to improve interoperability
- Consider simplified API for soft lock for legacy application, e.g. avoid need to introduce OO code there

Soft Lock Session Identifier

- For GUI/TTY Sessions, a GUID is suitable
- Authenticated /web requests receive a Session ID through the client-principal

Demo

- Review SmartLock API



```
/**
 * Purpose: Acquire logical application lock for a record
 * Notes:   This method is only implemented by SmartLockService
 *          Tries to create or update a SmartLock record
 * @param pcTableGuid Reference to unique record in SmartTable
 * @param pcKeyValues Values of a unique key field(s)
 * @param piLockDuration How long the lock may be hold in seconds
 * @param plThrowOnAlreadyLocked Logical indication to throw a record ... or not
 * @return True if lock is given else false
 */
METHOD PUBLIC LOGICAL AcquireLock (pcTableGuid AS CHARACTER,
                                   pcKeyValues AS CHARACTER,
                                   piLockDuration AS INTEGER,
                                   plThrowOnAlreadyLocked AS LOGICAL).
```

Consultingwerk
software architecture and development

Questions

Consultingwerk

software architecture and development