NOTES & DOMINO ENTWICKLERCAMP

assono

Fehlerbehandlung in Formelsprache, LotusScript, Java, JavaScript und XPages

EntwicklerCamp
13. März 2013

Innovative Software-Lösungen.

www.assono.de

NOTES & DOMINO ENTWICKLERCAMP

assono

Bernd Hort

Diplom-Informatiker, Universität Hamburg

seit 1995 entwickle ich Lotus Notes Anwendungen

IBM Certified Application Developer, System Administrator & Instructor

Sprecher auf diversen Konferenzen & Lotusphere 2008

bhort@assono.de
http://www.assono.de/blog
040/73 44 28-315

assono
IT-Consulting & Solutions

NOTES & DOMINO ENTWICKLERCAMP

assono

Agenda

- Motivation
- Formelsprache
- LotusScript
- Java
- JavaScript
- XPages
- OpenLog
- Fragen & Antworten

Motivation

Fehler passieren !!!

NOTES & DOMINO ENTWICKLERCAMP

assono

Strategie – Keine Fehlerbehandlung

Nicht wirklich zu empfehlen

Strategie – Fehler vermeiden

- Vorausschauend defensiv programmieren
  - Statt

```
Set doc = view.GetDocumentByKey("Key")
Messagebox doc.Title(0)
```

  - Besser

```
Set doc = view.GetDocumentByKey("Key")
If Not doc Is Nothing Then Messagebox doc.Title(0)
```

Strategie - Fehlerbehandlung

1. Fehlersituation erkennen
2. Versuchen den Fehler zu beheben
3. Wenn es nicht anders geht, den Benutzer informieren.

NOTES & DOMINO ENTWICKLERCAMP

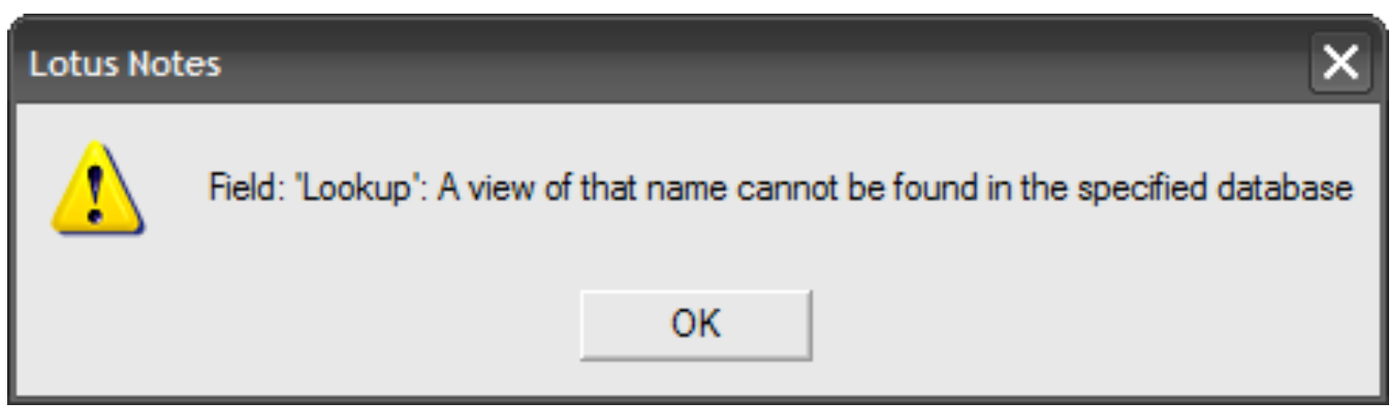
assono

Agenda

- Motivation
- Formelsprache
- LotusScript
- Java
- JavaScript
- XPages
- OpenLog
- Fragen & Antworten

Formelsprache – Allgemein

- Maske öffnet sich nicht, wenn ein Feld einen Fehler beinhaltet.



- Keine zentral definierte Fehlerhandling-Methode möglich
- Diverse Funktionen zum Testen

NOTES & DOMINO ENTWICKLERCAMP

assono

Formelsprache - @IsError

- Testet auf einen Fehler allgemein

```
_value:= @DBLookup(":"NoCache";""; "Ansicht"; "Schlüssel";2);
@If(@IsError(_value); "<Kein Eintrag gefunden>"; _value)
```

Formelsprache - @IfError

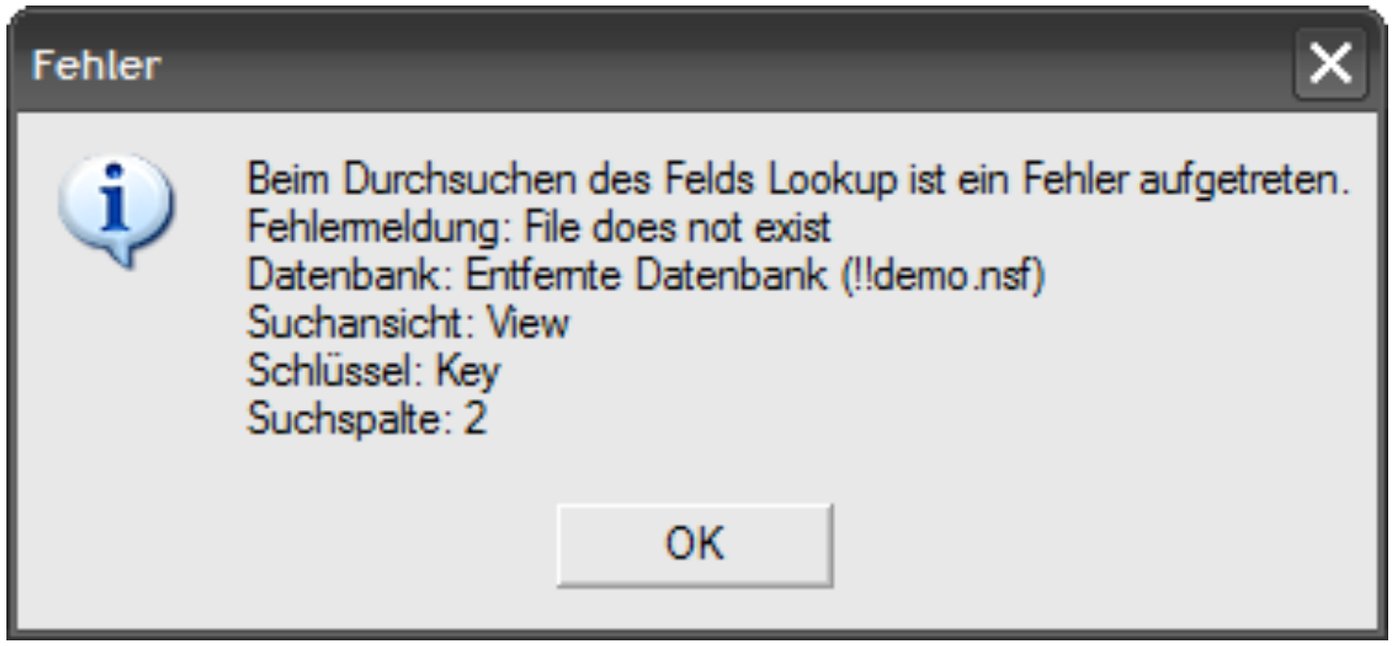
- Kombination aus If-Abfrage und Testen auf Error

```
@IfError(
   @DBLookup(""":"NoCache";""; "Ansicht"; "Schlüssel";2);
   "<Kein Eintrag gefunden>")
```

- Eingeführt in Version 6
- Als „deprecated“ in Version 7 gekennzeichnet
  - Auszug aus der Knowledgebase:
    @IfError does not work correctly when passed mathematical formulas or equations which result in an error.

Formelsprache – Lookup

- Mit entsprechendem Aufwand ist eine komfortable Fehlermeldung möglich.



```
_FieldLabel := "Lookup";
_DatabaseName := "Entfernte Datenbank";
REM "Instead of an ReplicaID also a Server:DBFilepath -Information is possible";
_ReplicaID := "":"demo.nsf";
_LookupView :=  "View";
_LookupKey := "Key";
_LookupColumn := 2;
_Cache := "NoCache";

REM "Far Lookup / BHT 2003-06-27";
REM "Placeholders";
REM "<LV> LookupView";
REM "<LC> LookupColumn";
REM "<LK> Lookupkey";
REM "<EM> ErrorMessage (from Notes)";
REM "<FL> FieldLabel";
REM "<CR> CarriageReturn";
REM "<DB> DatabaseName";
REM "<RI> ReplicaID";
```

```
_ErrorMessage := "Beim Durchsuchen des Felds <FL> ist ein Fehler aufgetreten.<CR>Fehlermeldung:
<EM> <CR>Datenbank: <DB> (<RI>)<CR>Suchansicht: <LV> <CR>Schlüssel: <LK> <CR>Suchspalte: <LC>";
_Placeholder:= "<FL>":"<EM>":"<CR>":"<LV>":"<LC>":"<LK>":"<DB>":"<RI>";

_Ret := @Unique(@DbLookup(""):_Cache; _ReplicaID; _LookupView; _LookupKey; _LookupColumn));
@If(@IsError(_Ret);
   @Do(
      @Set("_ErrorMessage";
               @ReplaceSubstring(_ErrorMessage; _Placeholder;

_FieldLabel:@Text(_Ret):@Char(13):_LookupView:@Text(_LookupColumn):_Lookupkey:_DatabaseName:@Imp
lode(_ReplicaID;"!!")
               )
               );
      @Prompt([OK]; "Fehler"; _ErrorMessage);
      @Return("<Kein Eintrag gefunden>"));
   _Ret);
```

NOTES & DOMINO ENTWICKLERCAMP

assono

@Funktionen zum Testen

- @IsNumber – Testen auf numerische Werte
- @IsTime – Testen auf Datums-/Zeitangaben
- @Elements – Anzahl der Listenelemente
- [...]

NOTES & DOMINO ENTWICKLERCAMP

assono

Agenda

- Motivation
- Formelsprache
- LotusScript
- Java
- JavaScript
- XPages
- OpenLog
- Fragen & Antworten

LotusScript – Fehlerbehandlung Basic

- On Error [errNumber] { GoTo label | Resume Next | GoTo 0 }
- [errNumber]
  - Optionale Angabe einer Fehlernummer
- GoTo label
  - Springt zur angegeben Sprungadresse
- Resume Next
  - Setzt die Abarbeitung in der nächsten Zeile fort
- GoTo 0
  - Kein Fehlerhandling

LotusScript - Fehlernummern

- Möglichkeit, je nach Fehlernummer unterschiedlich zu reagieren
- Fehlerkonstanten stehen in
  - lserr.lss – Allgemeine Fehler
  - lsxbeerr.lss – Notes spezifische Fehler
  - lsxuierr.lss – LSX Fehler
- Einige Fehlernummern sind allgemeinerer Natur
  - z.B. lsERR_NOTES_ERROR = 4000
  - Wird geworfen bei Abbruch durch Benutzer, Feld zu groß, Netzwerkprobleme etc.

LotusScript - Resume

- Resume [0 | Next | label]
- Setzt die Programmausführung fort
- Resume 0
  - Setzt die Programmausführung in der Zeile fort, in welcher der Fehler aufgetreten ist
- Resume Next
  - Setzt die Programmausführung in der nächsten Zeile fort
- Resume label
  - Setzt die Programmausführung ab der Sprungmarke fort

LotusScript - Fehlerinformationen

- Err – Fehlernummer
- Erl – Fehlerzeile
- Error – Fehlertext
- GetThreadInfo(InfoID) – Zeigt Informationen zum aktuellen Thread
  - LSI_THREAD_PROC – Name der aktuellen Procedure
  - LSI_THREAD_CALLPROC – Name der aufrufenden Precedure

NOTES & DOMINO ENTWICKLERCAMP

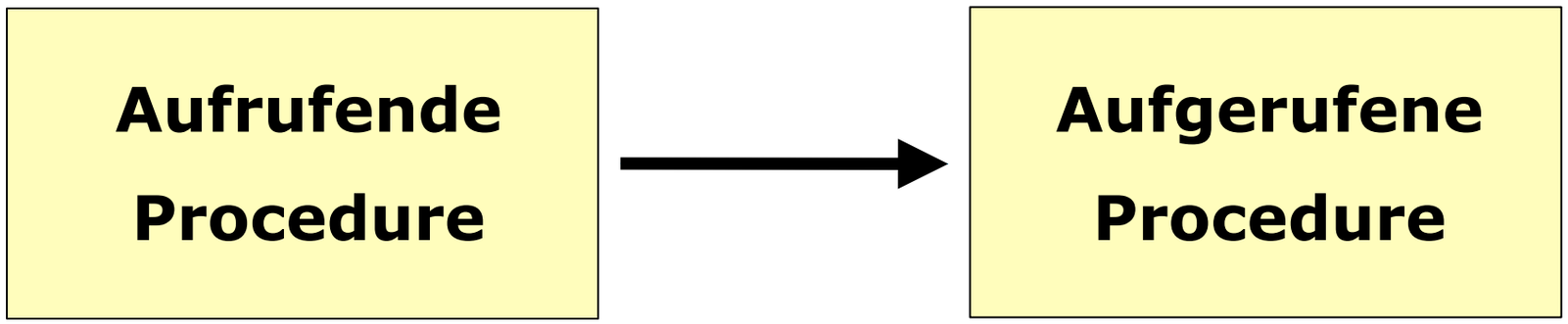
assono

LotusScript – LSI_Info

- Undokumentierte Funktion LSI_Info(InfoID)
- Vergleichbar mit GetThreadInfo(InfoID)
- Ab Version 6 liefert LSI_Info(14) den Stack aller aufgerufenen Proceduren
  - String, jeder Eintrag per Zeilenumburch getrennt
  - ThreadID, Name der Procedure, Zeilennummer
  - z.B.
    *54C7814,ERRORDEMO,11
    *54C7814,SUBPROCEDURE,4
    *54C7814,POSTOPEN,6
- Laut IBM nicht „thread safe“
  Achtung! Gefahr von Serverabstürzen!

Fehlerbehandlung bei kaskadierten Aufrufen

- Woher weiß die aufrufende Procedure, dass in der aufgerufenen Procedure ein Fehler passiert ist?



Fehlerbehandlung durch Rückgabewerte

- Der „klassische“ Ansatz
- Jede Procedure ist als Function definiert
- Zu Beginn der Function wird der Rückgabewert auf „False“ gesetzt.
- Erst kurz vor dem Ende der Function wird der Rückgabewert auf „True“ gesetzt.
- Die aufrufende Function prüft auf Rückgabewert, bevor sie weiter fortfährt.

Beispiel – Rückgabewerte

```
Function ErrorDemo() As Boolean
        On Error Goto err_handler

        ErrorDemo = False

        Messagebox doc.GetItemValue("Field")(0)
        Print "Ausgabe ohne Fehler"

        ErrorDemo = True
        Exit Function

err_handler:
        Messagebox "Es ist ein Fehler aufgetreten. Bitte informieren Sie Ihren Administrator." & _
        Chr$(10) & Chr$(10) & _
        "Procedure: " & Getthreadinfo(1) & Chr$(10) & _
        "Fehler-Nr.: " & Cstr(Err) & " / Zeile: " & Cstr(Erl) & Chr$(10) & _
        Error, MB_IconStop, "Fehler"

        Resume err_resume

err_resume:
End Function
```

Frisst den Fehler!!!

«Form»
LS Error Demo\Return values

Bewertung Fehlerbehandlung durch Rückgabewerte

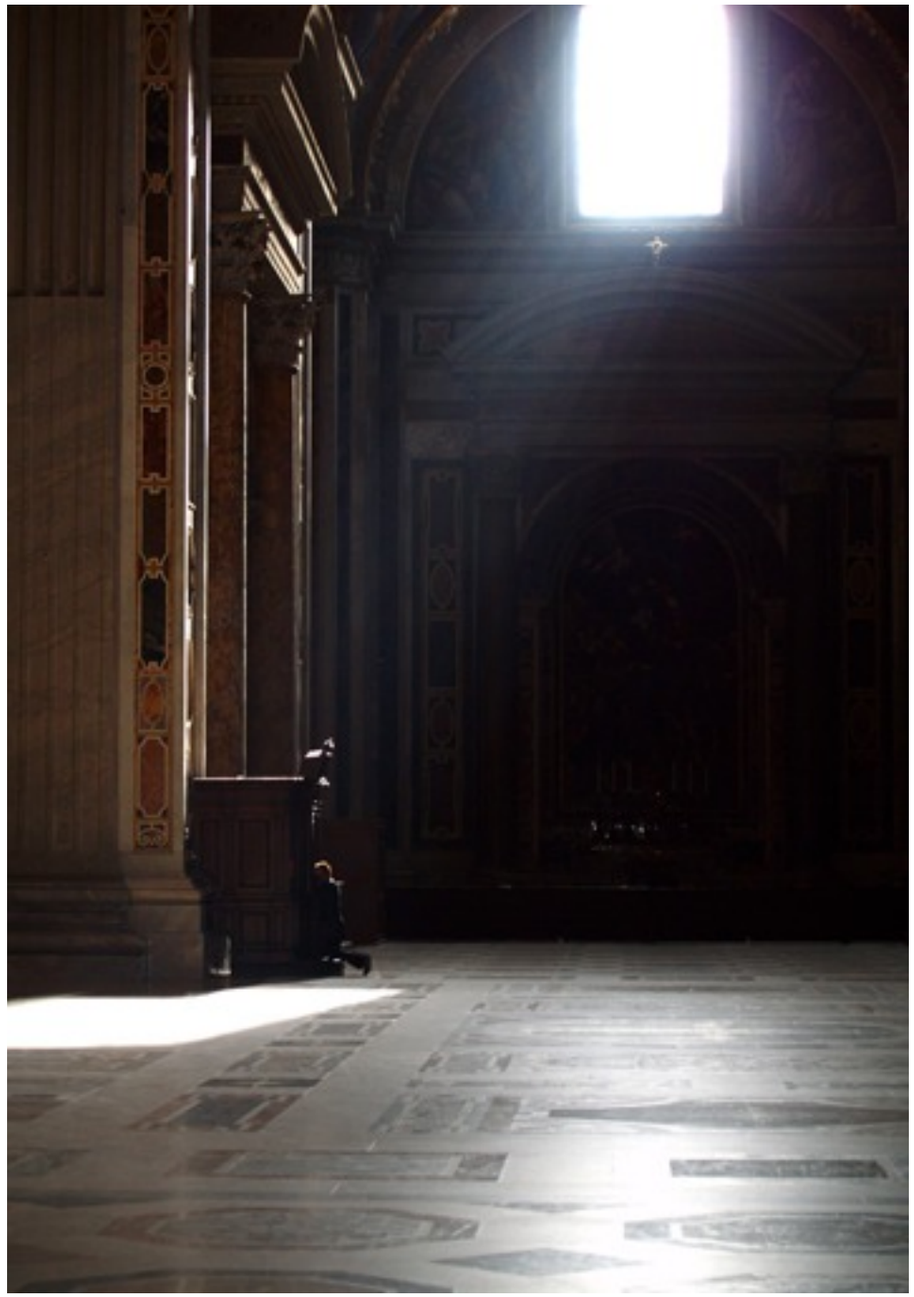
- Vorteile
  - Leicht zu implementieren
- Nachteile
  - Wenn die aufrufende Procedure nicht den Rückgabewert überprüft, wird die Ausführung trotz Fehler fortgesetzt!
  - Verschmutzt die Schnittstellen

NOTES & DOMINO ENTWICKLERCAMP

assono

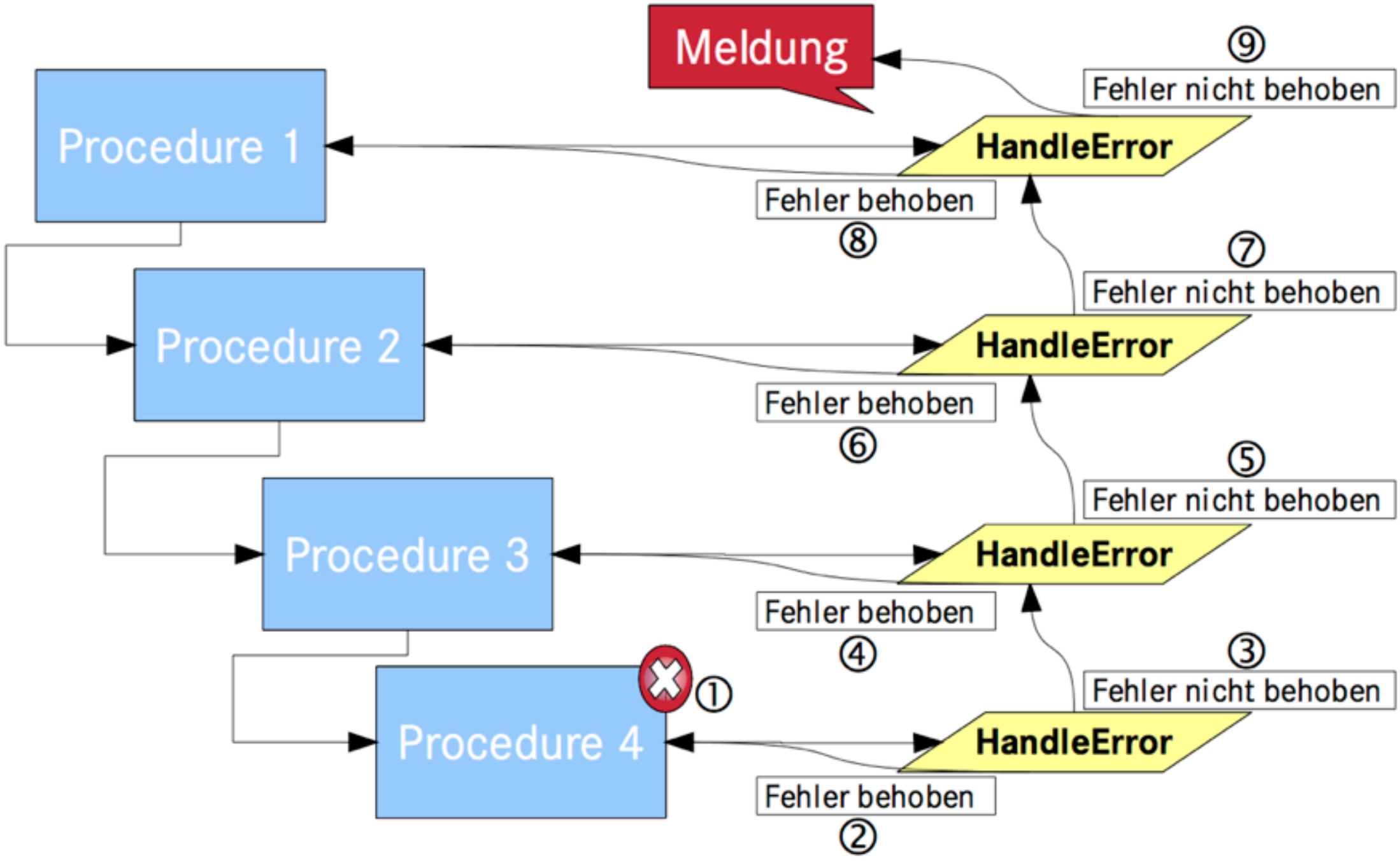
Geständnis

Ich war ein Schnittstellen-verschmutzer!!!



www.assono.de

Fehler nach oben durchreichen



«Form»
LS Error Demo\Error Chain

Hinweise

- Kein Resume
  - So lange der Fehler nicht wirklich behoben wurde
  - Oder die oberste Ebene erreicht wurde
- Im HandleError prüfen, ob oberste Ebene erreicht wurde
  - Getthreadinfo(LSI_THREAD_PROC) = _ Getthreadinfo(LSI_THREAD_CALLPROC))
- Mit End kann die gesamte Programmablauf beendet werden
- Ein Exit Function/Sub „frisst“ den Fehler ebenfalls

Selbst definierte Fehler

- Mit
  Error errNumber [, msgExpr]
  eigene Fehler werfen
- Sinnvoll errNumber > 10.000 zu wählen
- Sinnvoll Konstanten mit den Fehlernummern zu vergeben
  - Public Const ERRNO_DB_COULD_NOT_BE_OPENED% = 11001

Option Declare

- Jedesmal wenn LotusScript ohne „Option Declare“ gespeichert wird, stirbt ein kleines Kätzchen.
- Stellt sicher, dass alle Variablen vor ihrer Verwendung definiert worden sind.
- Kann ab Version 6 im Lotus Designer voreingestellt werden.

NOTES & DOMINO ENTWICKLERCAMP

assono

Agenda

- Motivation
- Formelsprache
- LotusScript
- Java
- JavaScript
- XPages
- OpenLog
- Fragen & Antworten

- Konzept der Exception
  - Eigene Fehlerklassen für unterschiedliche Fehlersituationen
  - Statt Fehlernummern zu unterscheiden werden Klassen unterschieden

- Try – Catch – Blöcke

```
try {
    // kritischer Code - Abschnitt
    PrintWriter pw = new PrintWriter (new FileWriter ("Ausgabe .txt"));
    pw. println (" Wert aus array : " + intarray [i]);
} catch ( IOException e) {
        // Code zur Fehlerbehandlung
} catch ( ArrayIndexOutOfBoundsException e) {
          // Code zur Fehlerbehandlung
} finally {
          // Aufräumarbeiten
}
```

Eigene Exception Klassen

- Ableitung von Exception
- Anreicherung der Fehlermeldung

```
// Eigene Exception
class KeywordNotFoundException extends Exception
{
    public KeywordNotFoundException ( String message ) {
        super ( message ); // Parameter wird an Elternklasse gegeben
    }
}
```

Hochreichen von Fehlern

- Neue Runtime-Exception werfen, wenn die Methode selber den Fehler nicht handeln kann.

```
try {
      someRiskyMethode();

} catch (Exception e) {
    throw new RuntimeException(e);
}
```

- Nur sinnvoll bei unerwarteten Fehlern.
- Mit exception.getCause() kann auf das ursprüngliche Fehlerobjekt zurückgegriffen werden.

NotesException und NotesError

- NotesException hat zwei public Variablen
  - id - gibt die Notes Fehlernummer aus
  - text - gibt den Notes Fehlertext aus
- NotesError ist ein Interface für die Notes Fehlernummern

```
try {
    Database db = CommonUtils.getCurrentDatabase();
    View view = db.getView(VIEW_NAME);
    if (view == null) {
        throw new NotesViewNotFoundException(VIEW_NAME);
    }
} catch (NotesException ne) {
    //Prüft ob der Fehler durch ein ungültiges Dokument verursacht wird
    if (ne.id==NotesError.NOTES_ERR_INVALID_DOC) {

    }
} catch (Exception e) {
    throw new RuntimeException(e);
}
```

«Agent»
Java Error

NOTES & DOMINO ENTWICKLERCAMP

assono

Agenda

- Motivation
- Formelsprache
- LotusScript
- Java
- JavaScript
- XPages
- OpenLog
- Fragen & Antworten

JavaScript

- Mit throw Bezeichnung wird ein Fehler geworfen
- Try – Catch – Blöcke

```
function getMonthName (mo) {
    mo=mo-1; // Adjust month number for array index (1=Jan, 12=Dec)
    var months=new Array("Jan","Feb","Mar","Apr","May","Jun","Jul",
        "Aug","Sep","Oct","Nov","Dec");
    if (months[mo] != null) {
       return months[mo]
    } else {
       throw "InvalidMonthNo"
    }
}

try {
   // statements to try
    monthName=getMonthName(myMonth) // function could throw exception
}
catch (e) {
    monthName="unknown"
    logMyErrors(e) // pass exception object to error handler
}
```

Mehrere Catch – Blöcke

```
try {
    // function could throw three exceptions
        getCustInfo("Lee", 1234, "lee@netscape.com")
}
catch (e if e == "InvalidNameException") {
    // call handler for invalid names
        bad_name_handler(e)
    }
catch (e if e == "InvalidIdException") {
    // call handler for invalid ids
        bad_id_handler(e)
    }
catch (e if e == "InvalidEmailException") {
    // call handler for invalid email addresses
        bad_email_handler(e)
    }
catch (e){
    // don't know what to do, but log it
        logError(e)
    }
```

Onerror Event Handler

- Ein zentraler Event Handler für Fehler kann registriert werden
  - window.onerror = handler-func
- Der Browser ruft die Funktion auf mit
  - window.onerror(message, url, line)
  - Message ist die Fehlermeldung
  - Url ist die aktuelle Seite
  - Line ist die Zeile
- Rückgabewert
  - Bei True wurde der Fehler behandelt
  - Bei False zeigt der Browser eine Standardfehlermeldung an

«Form»
Demo JavaScript Error

NOTES & DOMINO ENTWICKLERCAMP

assono

Agenda

- Motivation
- Formelsprache
- LotusScript
- Java
- JavaScript
- XPages
- OpenLog
- Fragen & Antworten

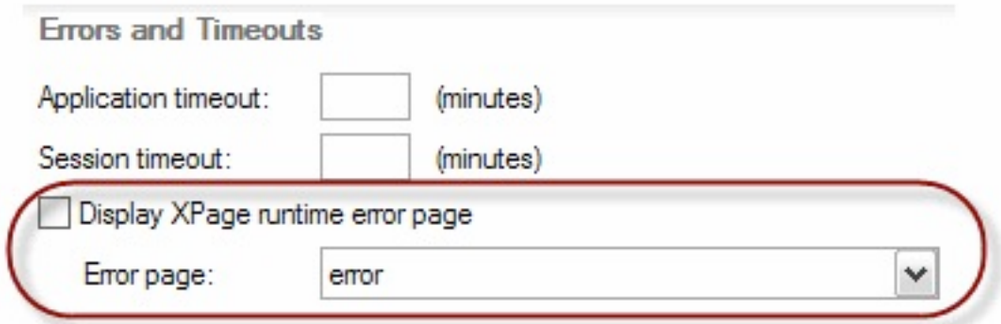
In den Programmiersprachen

- Prinzipiell gelten in XPages die vorgestellten Mechanismen wie z.B. try-catch-Blöcke für
  - ServerSide JavaScript
  - ClientSide JavaScript
  - Java

- In ServerSide JavaScript helfen die Methoden
  - print(text) - Ausgabe von Text auf der Server Konsole
  - _dump(object) - Ausgabe aller Eingeschaften eines Objectes auf der Server Konsole

Eigene Fehlerseite

- Fehlerverhalten für XPages wird über die XPages Properties innerhalb der Application Properties festgelegt



- Eigene Error Page ist „anwenderfreundlicher“
  - Wird nur beim Full-Refresh angezeigt
  - Bei Partial-Refresh erscheint ein Fehlerdialog
- „Display XPage runtime error page“ zeigt immer detailierte Fehlerinformationen

Eigene Fehlerseite

- Fehler kann über **`requestScope`**`.error` ausgelesen werden.
- Rückgabewert ist ein Exception Object

```
try{
    var stackTrace ="";
    var error = requestScope.error;
    var errorMessage = "";
    if (error!=null){
        var trace = error.getStackTrace();
        errorMessage = '<div style="color:red;margin-bottom:5px;">' +error.toString() + '</div>\n';
        while(typeof(error)!="NotesException" && error.getCause()!=null){
                trace = error.getStackTrace();
                error = error.getCause();
                errorMessage += '<div style="color:red;margin-bottom:5px;">caused by '
+error.toString() + '</div>\n';
            }

        for( var i = 0; i < trace.length; i++){
            stackTrace += trace[i] + '<br>\n';
        }
        return errorMessage + '<div style="color:gray;margin-bottom:5px;">' + stackTrace + '</div>';
    } else {
        return "No error object found";
    }
} catch(e){
    print(e.toString());
}
```

NOTES & DOMINO ENTWICKLERCAMP

assono

EventHandler onError

- Im EventHandler kann ClientSide JavaScript-Code definiert werden, der im Fehlerfall aufgerufen werden soll.

```
<xp:eventHandler
        event="onclick"
        submit="true"
        refreshMode="partial"
        refreshId="panel1">
        <xp:this.action><![CDATA[#{javascript:thereIsNoSuchFunction();}]]>
        </xp:this.action>
        <xp:this.onError><![CDATA[console.log(arguments);
        XSP.error("Es ist ein Fehler aufgetreten.\nBitte informieren Sie
        den Benutzerservice.");]]></xp:this.onError>
</xp:eventHandler>
```

«XPage»
Hello World

Hinweise für die Fehlersuche

- Auf dem Server liegt eine error-log-0.xml Datei im Verzeichnis {Datenpfad}/domino/workspace/logs
- OpenNTF Project XPages Log File Reader
  http://www.openntf.org/internal/home.nsf/project.xsp?action=openDocument&name=XPages%20Log%20File%20Reader
  - Zugriff auf verschiedene Log-Datei via XPage
- OpenNTF Project XPages Toolbox zum Profiling
  http://www.openntf.org/internal/home.nsf/project.xsp?action=openDocument&name=xpages%20toolbox
  - Sollte nicht in Produktion eingesetzt werden
- Eclipse Memory Analyser
  - Separates Tool um Speicherdumps auszuwerten

NOTES & DOMINO ENTWICKLERCAMP

assono

Agenda

- Motivation
- Formelsprache
- LotusScript
- Java
- JavaScript
- XPages
- OpenLog
- Fragen & Antworten

NOTES & DOMINO ENTWICKLERCAMP

assono

OpenNTF – OpenLog

- Open Source Lösung zum zentrallen Loggen von Fehlern
- Unterstützung für
  - LotusScript
  - Java
  - JavaScript
- Einfache Integration in bestehenden Code

NOTES & DOMINO ENTWICKLERCAMP

assono

LogError und LogErrorEx

- Function LogError() As String
  - Einfaches Loggen
  - Gibt Standardfehlermeldung zurück
- Function LogErrorEx (msg As String, severity As String, _ doc As NotesDocument) As String
  - Individuelle Fehlermeldung
  - Schwere des Fehlers
  - Dokumentverknüpfung

Demo OpenLog

NOTES & DOMINO ENTWICKLERCAMP

assono

Agenda

- Motivation
- Formelsprache
- LotusScript
- Java
- JavaScript
- XPages
- OpenLog
- Fragen & Antworten

NOTES & DOMINO ENTWICKLERCAMP

assono

Fragen?

jetzt stellen – oder später:

bhort@assono.de

http://www.assono.de/blog

040/73 44 28-315

assono
IT-Consulting & Solutions

Folien & Beispieldatenbank unter
http://www.assono.de/blog/d6plinks/EntwicklerCamp-2013-Fehlerbehandlung