Wir entwickeln Ihr Potenzial.

assono

Fehlerbehandlung in Formelsprache, LotusScript, Java und JavaScript

Softsphere
Admin & Developer

Bernd Hort
assono GmbH
bhort@assono.de
http://www.assono.de
+49/4101/48747

Agenda

- Vorstellung
- Motivation
- Formelsprache
- LotusScript
- Java
- JavaScript
- OpenLog
- Zusammenfassung

Vorstellung

- Bernd Hort
- Diplom-Informatiker
- Lotus Notes Anwendungsentwicklung seit 1995
- IBM Certified Application Developer - Lotus Notes and Domino 7
- IBM Certified System Administrator –Lotus Notes and Domino 7
- IBM Certified Instructor SA & AD –Lotus Notes and Domino 7
- Sprecher Lotusphere 2008

Motivation

Fehler passieren!!!

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.

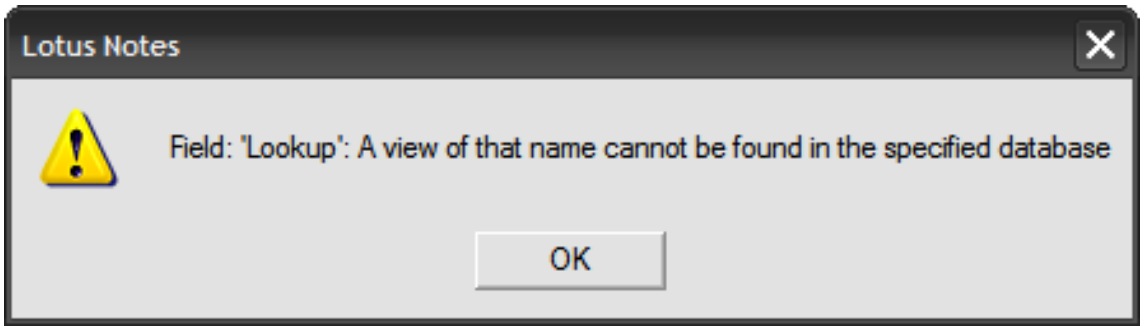
assono

Agenda

- Vorstellung
- Motivation
- Formelsprache
- LotusScript
- Java
- JavaScript
- OpenLog
- Zusammenfassung

Formelsprache – Allgemein

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



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

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

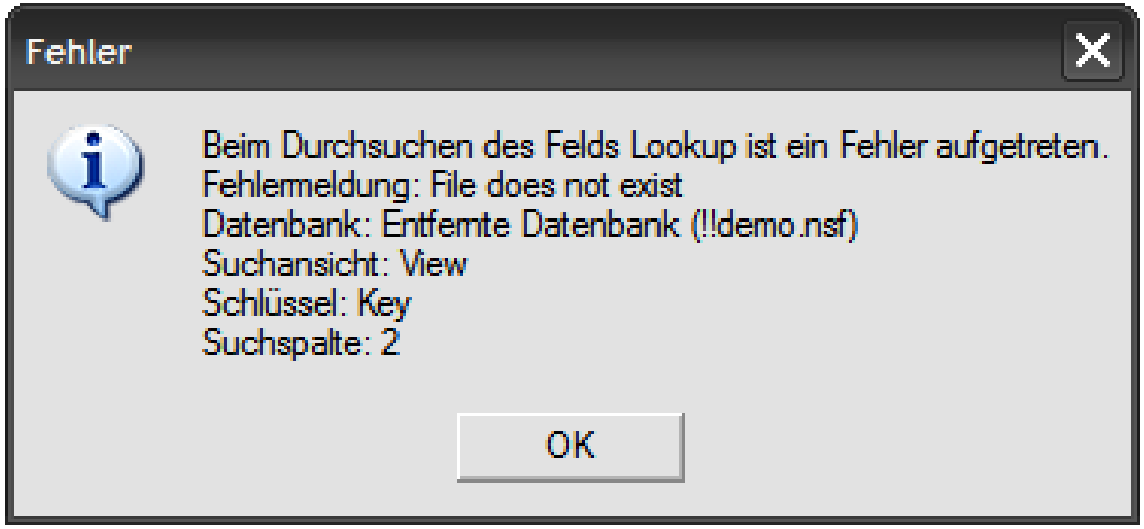
```
@IfError(
    @DBLookup(""":"NoCache";""; "Ansicht"; "Schlüssel";2);
    "<Kein Eintrag gefunden>")
```

- 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:@Implode(_ReplicaID;"!!")
          )
          );
    @Prompt([OK]; "Fehler"; _ErrorMessage);
    @Return("<Kein Eintrag gefunden>"));
  _Ret);
```

@Funktionen zum Testen

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

Agenda

- Vorstellung
- Motivation
- Formelsprache
- LotusScript
- Java
- JavaScript
- OpenLog
- Zusammenfassung

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

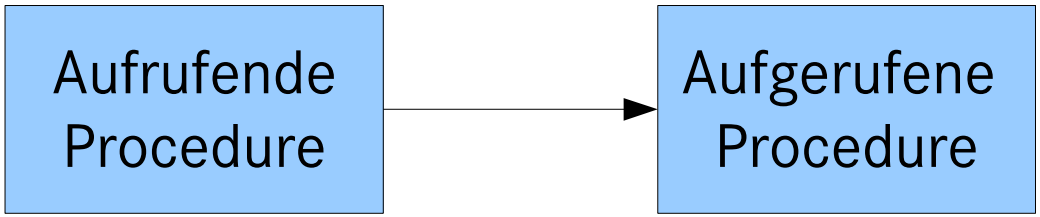
- Err – Fehlernummer
- Erl – Fehlerzeile
- Error – Fehlertext
- GetThreadInfo(InfoID) – Zeigt Informationen zum aktuellen Thread
  - LSI_THREAD_PROC – Name der aktuellen Procedure
  - LSI_THREAD_CALLPROC – Name der aufrufenden Precedure

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! Kann zu Serverabstürzen führen!

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!!!

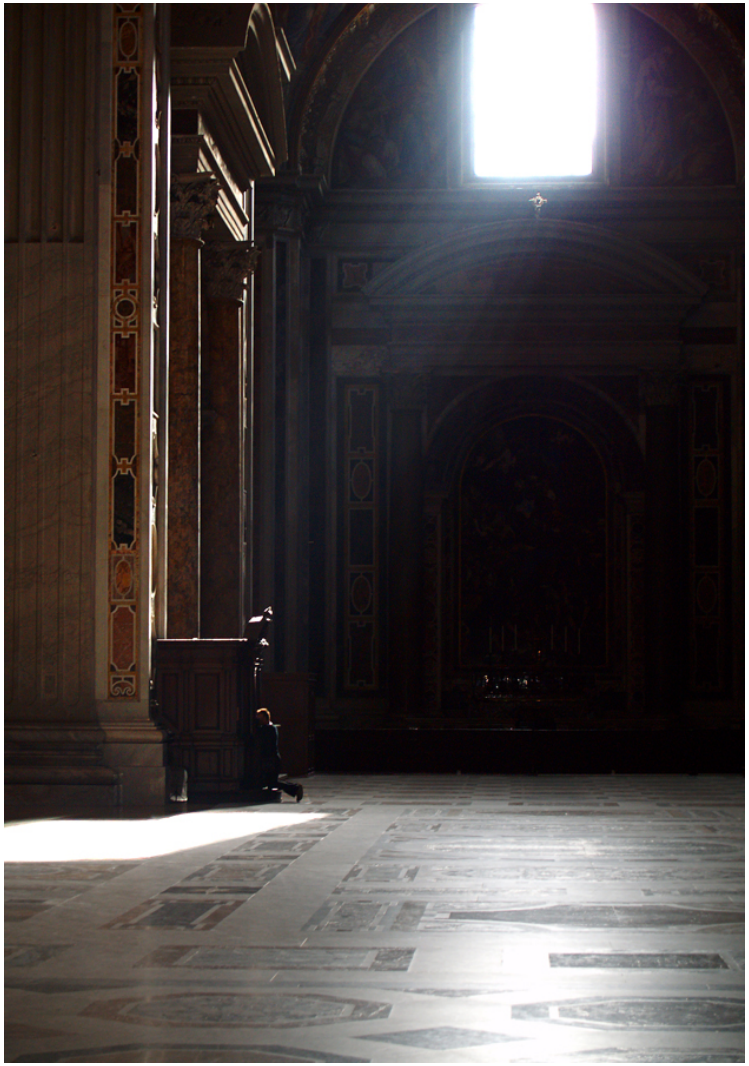
«Form»
LS Error Demo\Return values

Bewertung
Fehlerbehandlung durch Rückgabewerte

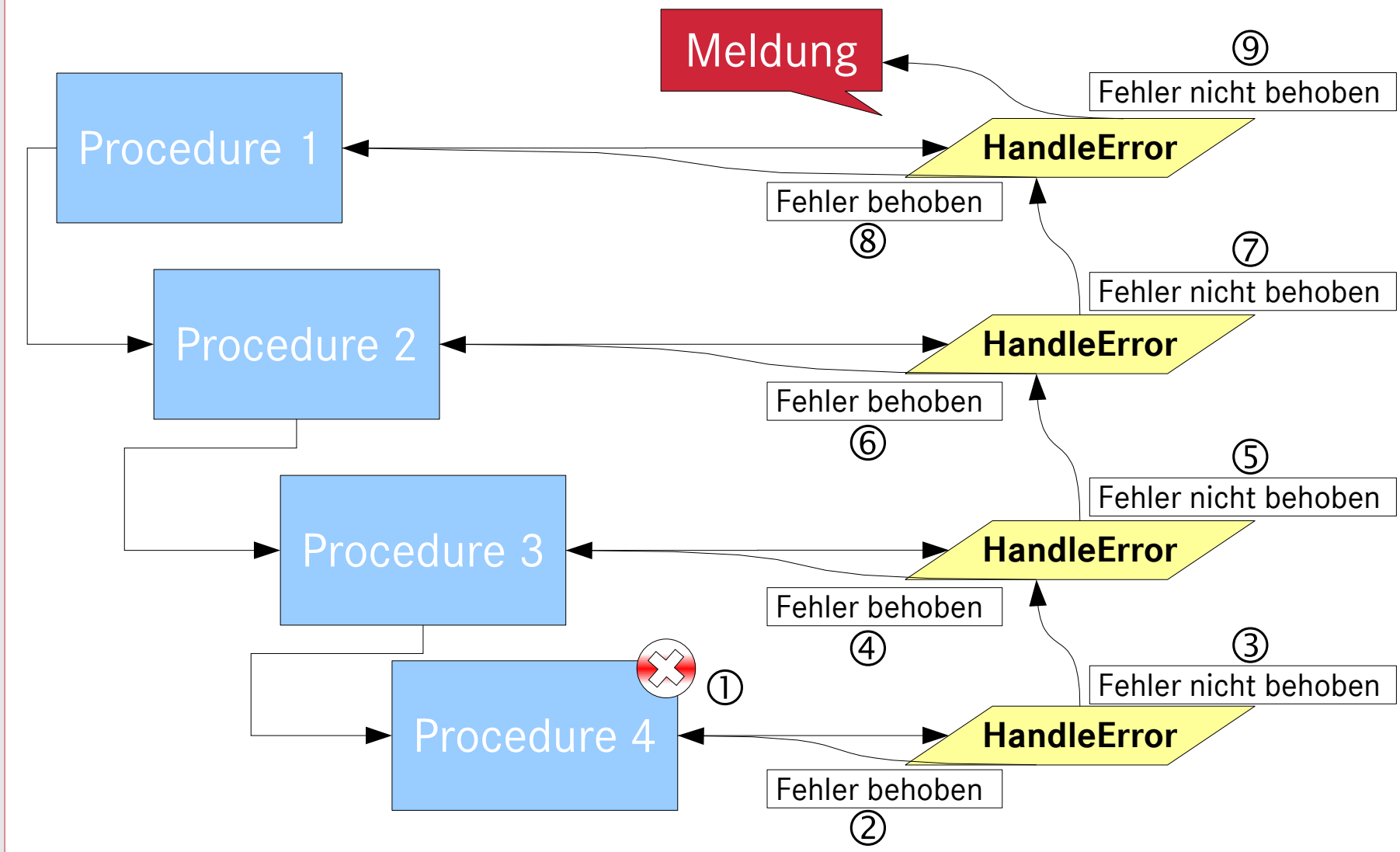
- Vorteile
  - Leicht zu implementieren
- Nachteile
  - Wenn die aufrufende Procedure nicht den Rückgabewert überprüft, wird die Ausführung trotz Fehler fortgesetzt!
  - Verschmutzt die Schnittstellen

Geständnis

Ich war ein Schnittstellen-verschmutzer!!!



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 Sub/Function „frisst“ einen Fehler

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.

Agenda

- Vorstellung
- Motivation
- Formelsprache
- LotusScript
- Java
- JavaScript
- OpenLog
- Zusammenfassung

Java

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

```
try {
        // kritischer Code - Abschnitt
        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
        }
}
```

«Agent»
Java Error

Agenda

- Vorstellung
- Motivation
- Formelsprache
- LotusScript
- Java
- JavaScript
- OpenLog
- Zusammenfassung

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

Agenda

- Vorstellung
- Motivation
- Formelsprache
- LotusScript
- Java
- JavaScript
- OpenLog
- Zusammenfassung

OpenNTF – OpenLog

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

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

Agenda

- Vorstellung
- Motivation
- Formelsprache
- LotusScript
- Java
- JavaScript
- OpenLog
- Zusammenfassung

Fragen ?

- **Fragen?**
  - jetzt stellen oder später
    - E-Mail: bhort@assono.de
    - Telefon: 04101 / 487 47
    -
- Folien und Beispiele unter http://www.assono.de/blog.nsf/d6plinks/Softsphere2008