Fehlerhandling in Formelsprache, Lotus Script und Java

EntwicklerCamp 2006 20.2.-22.2.2006

Reinhard Theis
IT-Architekt
holistic-net GmbH

Goal

Es geht in dieser Session nicht darum, wie ich die meisten Informationen heraushole, sondern um die Philosphie der sinnvollen Realisierung in der Praxis in Notes/Domino Projekten, insbesondere beim Abarbeiten von grossen Datenmengen durch Agenten

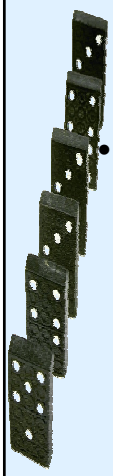
The Power of Notes & Domino at your Grasp
Powered by Bundled-KnowHow

Agenda

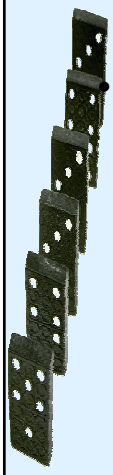
- Einleitung und Leitgedanken
- @Formelsprache
- LotusScript
- Java

Motivation

- Fehlervermeidung und effiziente Erkennung
- Die Ursachenforschung für Probleme zu minimieren
- Aber: Vermeidung unnötigen Log-Volumens, dass eh keiner auswerten kann



- Ein Agent, der 10000 Dokumente nachts bearbeiten soll, soll nicht seine Arbeit aufgeben, weil beim 5. Dokument ein Problem auftrat!



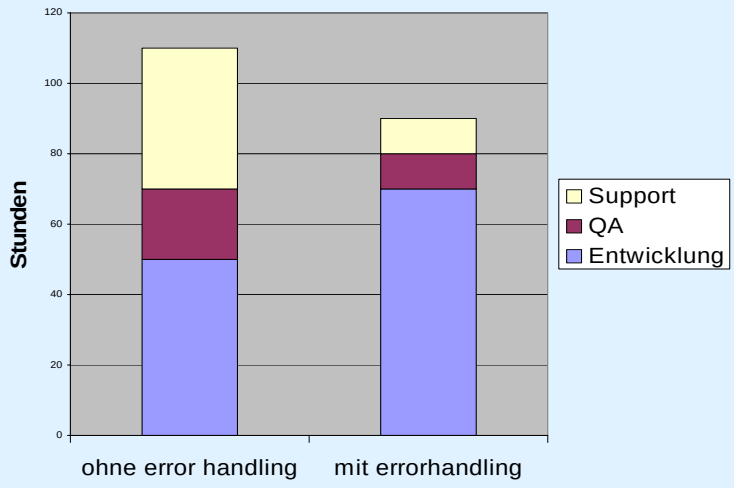
Handicap

- Die Realisierung eines effektiven Fehlerhandling ist mit viel und vor allem stupider Entwicklungsarbeit verbunden
  - Hier hilft oft nur Disziplin!
  - Erhöht den Realisierungsaufwand

Aber ..

- Später zahlt sich der Aufwand auf
  - Schnelle Ursachenerkennung
  - Verringerung des Supportaufwandes (bzw. des Gewährleistungsaufwandes)

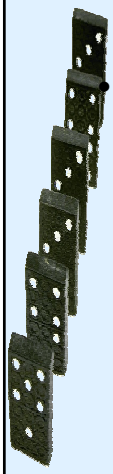
Aufwände mit und ohne....



B-KH

EntwicklerCamp '06

In Media res...

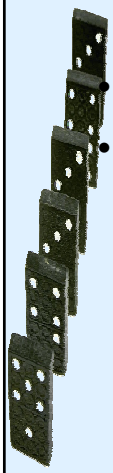


- Moderne Betriebssysteme/Middleware ist in der Regel vorbereitet, dass Prozesse die in ihm gestartet werden und auf Fehler laufen, sauber enden und nicht die Laufzeitumgebung zum Absturz bringen
  - Anspruch und Realität?
  - Aber: Wenn wir Agenten schreiben, wollen wir die Kontrolle auch im Fehlerfalle behalten!

B-KH

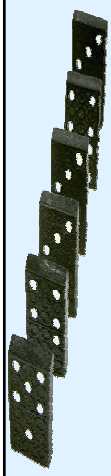
EntwicklerCamp '06

Ansprüche

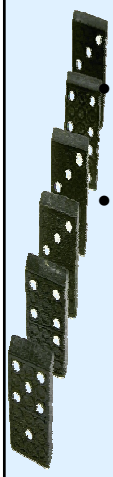


- kurze Analysezeiten im Problemfalle
- Kontrolle behalten über den Prozess
  - Wir wollen entscheiden was im Fehlerfalle passiert
    - Dokumentation (logging)
    - Entscheidung wie der Prozess fortgesetzt wird

The Power of Notes & Domino at your Grasp
Powered by Bundled-KnowHow



@Formelsprache



@Formelsprache

- Wichtig: Vermeidung von Abspeicherung von Fehlertexten in Feldern.
- Häufigste Fehlerquelle: @dblookup oder @dbcolumn
  - Liefern auch einen Fehler zurück, wenn keine Elemente gefunden!

B-KH

EntwicklerCamp '06

- Wichtigste Hilfe:

  @IsError

Dateitypen überprüfungen in Zusammenhang mit @IsTime und @IsNumber

B-KH

EntwicklerCamp '06

- Demo

The Power of Notes & Domino at your Grasp

Powered by Bundled-KnowHow

B-KH

EntwicklerCamp '06

Lotusscript

The Power of Notes & Domino at your Grasp

Powered by Bundled-KnowHow

B-KH

EntwicklerCamp '06

Lotus Script

- On error statement
  - Angabe eines Sprungzieles im Falle eines Fehlers
    - On error goto SysErrorExit
      .....code
      exit sub/function
      Syserrorexit:
      .....code der im Fehlerfalle ausgeführt wird
      exit sub/function
  - Kann auf spezielle Fehler reduizert werden
    - On error <Fehlernummer>

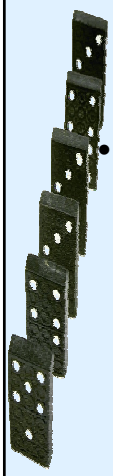
The Power of Notes & Domino at your Grasp

Powered by Bundled-KnowHow

B-KH

EntwicklerCamp '06

Worry..



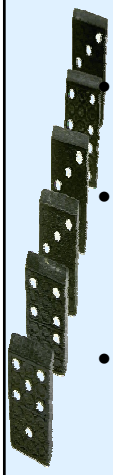
- DAS sollte in keinem Code ohne ausführlicher Begründung stehen:

  On error resume next

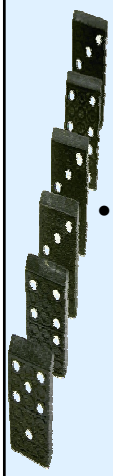
B-KH

EntwicklerCamp '06

LS- Funktionen

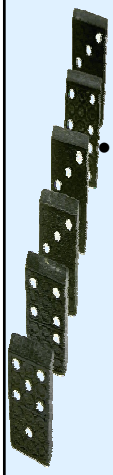


- Err
  Gibt die Fehlernummer zurück
- Erl
  Gibt die Zeile zurück, in der der Fehler aufegtreten ist
- Error (Err)
  Gibt den Fehler als Text zurück



- Error Statement zum Erzeugen eigener Fehlerevents
  - -“Werfen von Fehlerevents“
  - Error 99,“Mein Fehler“

Weitere hilfreiche Funktionen



- GetThreadInfo(<infoID>)
  - <infoID> Konstanten, die in der Datei LSPVars.lss definiert sind (%INCLUDE "LSConst.LSS)

Konstanten für getTrheadInfo

LSI_THREAD_LINE	Current Line Number
LSI_THREAD_PROC	Name of current procedure
LSI_THREAD_MODULE	Name of current module
LSI_THREAD_VERSION	LotusScript version number
LSI_THREAD_LANGUAGE	(Human) language setting
LSI_THREAD_COUNTRY	Country or region setting
LSI_THREAD_TICKS	Get current clock ticks
LSI_THREAD_PROCESS_ID	Get current process ID
LSI_THREAD_TASK_ID	Get current task ID
LSI_THREAD_CALLPROC	Get the name of the calling procedure
LSI_THREAD_CALLMODULE	Get the name of the calling module

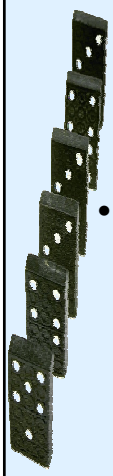
undokumentiert

- LSIInfo(1) Current Linenumber
- LSIInfo(2) Current Procedure
- LSIInfo(3) Current Module
- LSIInfo(6) Lotusscript Version
- LSIInfo(9) Language Settings
- LSIInfo(12) Calling Procedure
- LSIInfo(50) LS Memory allocated
- LSIInfo(51) OS Memory allocated
- LSIInfo(52) LS Blocks used

B-KH

EntwicklerCamp '06

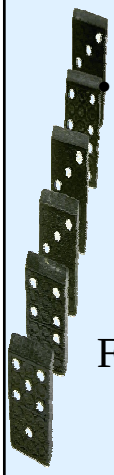
Demo



- Demo wir entwicklen Step bei Step

B-KH

EntwicklerCamp '06

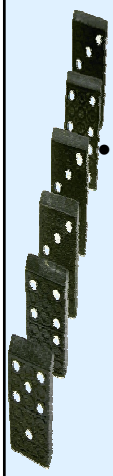


- Do not „handle“ errors from your code, only „throw“ them. Let the calling code (client) worry about the „handling“ the error.

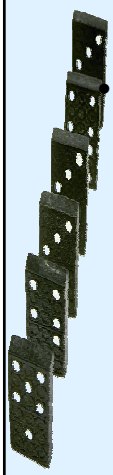
Frage: wo fängt der „calling code“ an? Ist das erst die Notes-Client/Server oder noch mein eigener Agent?

The Power of Notes & Domino at your Grasp

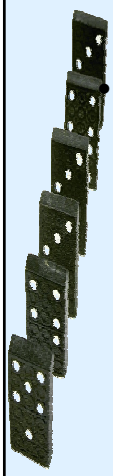
Powered by Bundled-KnowHow



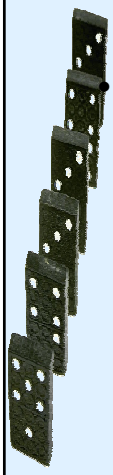
- Trennung von Prozess und Funktion
  - Der Prozess muss entscheiden, was im Falle eines Fehlers in einer Funktion geschehen soll.
    - Logging
    - Entscheidung wie weiter



- Konsequenz:
  - Keine Ausgabe der Fehlermeldung in der Funktion (print/messagebox o.ä.)
  - Voraussetzung auch zur Wiederverwertbarkeit einer Funktion und Nutzung sowohl in Front-End als auch Backend Prozessen



- In der Praxis hat sich selbstprogrammierte Fehlerweitergabe und nicht das „werfen“ eines Fehlers nach oben bewährt
  - Jede Funktion gibt einen Intergerwert zurück (0=ohne Fehler) sowie als letzten Parameter einen Fehlertext.
  - Nachteil: Die Zeilennummer erhält man nicht mehr mit
    - Nachteil ist aber bei konsequenter Modularisierung und Kleinhaltung des einzelnen Funktionscodes vertretbar



Best Practices

- Nicht nur auf den LS-Systemfehlerhandler bauen!
  - LS Systemfehlermeldungen helfen oft nicht sofort weiter oder ausschließlich dem Entwickler

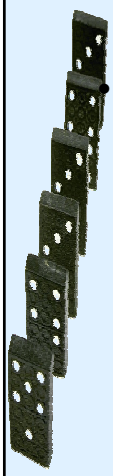
    Soviel wie möglich aussagekräftige Fehlermeldungen erzeugen!

  - Beispiel: Ansicht existiert nicht (oder keine Zugriffsrechte)

B-KH

EntwicklerCamp '06

Best Practices



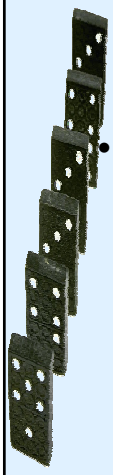
- Wenden Sie diese Vorgehensweise auf alle(!) Funktionen an
  - Ausnahmen nur einfachste Utility Funktionen

Es Lohnt sich!

B-KH

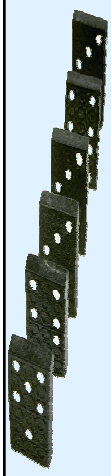
EntwicklerCamp '06

Best Practices

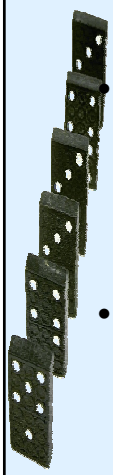


- Eigenes Logging bewahrt Administrator vor überfrachtetem notes.log
  - Nutzung einer eigenen Logklasse ermöglicht individuelle Gestaltung und zusätzliche Felder!

The Power of Notes & Domino at your Grasp
Powered by Bundled-KnowHow



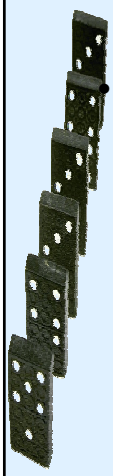
Java



Java

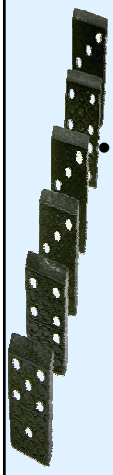
- Java kann neben LS als Programmiersprache für Agenten in Notes/Domino Umgebungen eingesetzt werden.
- Java ist ausschließlich Objektorientiert!

Differenzen zu Lotus Script

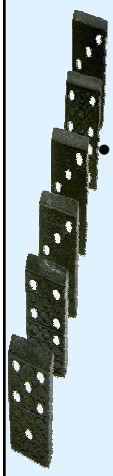


- Exception Konzept
  - Im Prinzip ähnlicher Eventhandler wie bei Lotus Script
  - Gekapselt in der sog. Exception Class
  - Möglichkeit, eigene zu schreiben
    - Es stehen alle Möglichkeiten offen, daher wesentlich mächtiger als der einfache Fehlerevent in LS

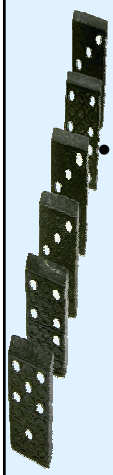
Vorteil gegenüber LS



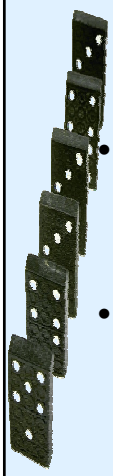
- Es gibt das Konstrukt try{} catch{} !!
  - Ermöglicht, „geworfene“ Exceptions gezielt und abzufangen
  - Funktionen können so geschrieben werden, dass aufrufende Funktion evtl. geworfene Exception abgefangen muss!



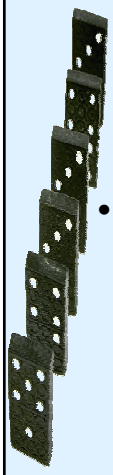
- Hurra, wir brauchen nicht das hochgeben nachprogrammieren, um gezielt und kontrolliert auszuwerten!



- Voraussetzung: Die Funktion wirft jeden Fehler als Exceptions.
  - Auch wenn Mehrarbeit, schreiben Sie für jede Klasse einen eigenen Exception-Klasse



- Machen Sie Gebrauch davon! Zwingen Sie den Nutzer einer Funktion zum Auswertung einer Exception
- Auch hier gilt: In Funktionen keine Ausgabe machen, sondern Exceptions werfen

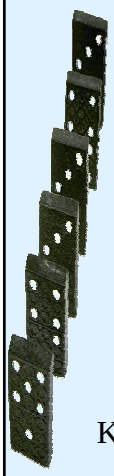


- Demo

B-KH

EntwicklerCamp '06

Lotus Notes



Q & A

Kontakt: reinhard.theis@holistic-net.de

The Power of Notes & Domino at your Grasp

Powered by Bundled-KnowHow