

Entwicklung einer Java Add-In Servertask



ABData
IT Consulting and Engineering

Andy Brunner
Kappelstrasse 43
CH-8002 Zürich



Andy.Brunner@ABData.CH

notesnet.ch

Bilderrätsel



=



+



=



!

Zu meiner Person

- Seit 34 Jahren in der IT Branche.
Quizfragen: Was sind Lochkarten, EDV, Telex, cc:Mail, OS/2, SNA, 3270, NetBIOS, 300-Baud Acoustic Coupler, MASM und REXX?
- Einzelfirma für «IT Consulting and Engineering» in den Bereichen Internet, Kommunikation, Security, Domino, Linux und Java
- «Yellow-Bleeder» seit Notes 4
- IBM Notes/Domino Design Partner
- Gründungsmitglied der NotesNet.CH Vereinigung (Kleine Gruppe von Notes/Domino Business Partnern in der Schweiz)

Einsatzgebiete von Domino Add-Ins

- Zum Erstellen von systemnahen Domino Serverprozessen
- Zugriffe auf alle Serverfunktionen (läuft unter der ID des Servers)
- Für ad-hoc oder dauernd laufende Systemtasks
- Kein User-Interface – Nur Meldungen auf der Domino Konsole
- Nach Möglichkeit für alle Domino Plattformen lauffähig machen
- Traditionell mit C++ entwickelt, immer häufiger jedoch auch mit Java (z.B. iSpy, RMEval, ChangeMan)

Java VM Versionen

- Domino enthält eine IBM Java JVM (z.B. Win32: ...\\IBM\\Lotus\\Domino\\jvm)
- Sehr nahe kompatibel mit entsprechenden Sun JVM Versionen
- Java Versionen vor 1.4 (Java 4) sollten nicht mehr verwendet werden

Domino Version	IBM JVM Version
8.5	1.6.0 (Java 6)
8.0	1.5.0 (Java 5)
7.0	1.4.2 (Java 4)
6.5/6.0	1.3.1 (Win32, Linux, Solaris) 1.1.8 (zOS, zLinux, iSeries)

Überprüfung Java JVM Version

- Ab Version 8 kann auf der Domino Konsole «Show JVM» eingegeben werden, um die JVM Version anzuzeigen:

> Show JVM

10.02.2009 16:35:59 JVM: Java Virtual Machine initialized.

10.02.2009 16:35:59 JVM version: J2RE 1.6.0 IBM J9 2.4 Windows XP x86-32 jvmwi3260ifx-20081002_23977 (JIT enabled, AOT enabled)

J9VM - 20081002_023977_IHdSMr JIT - r9_20080415_1520ifx7 GC - 20080415_AA

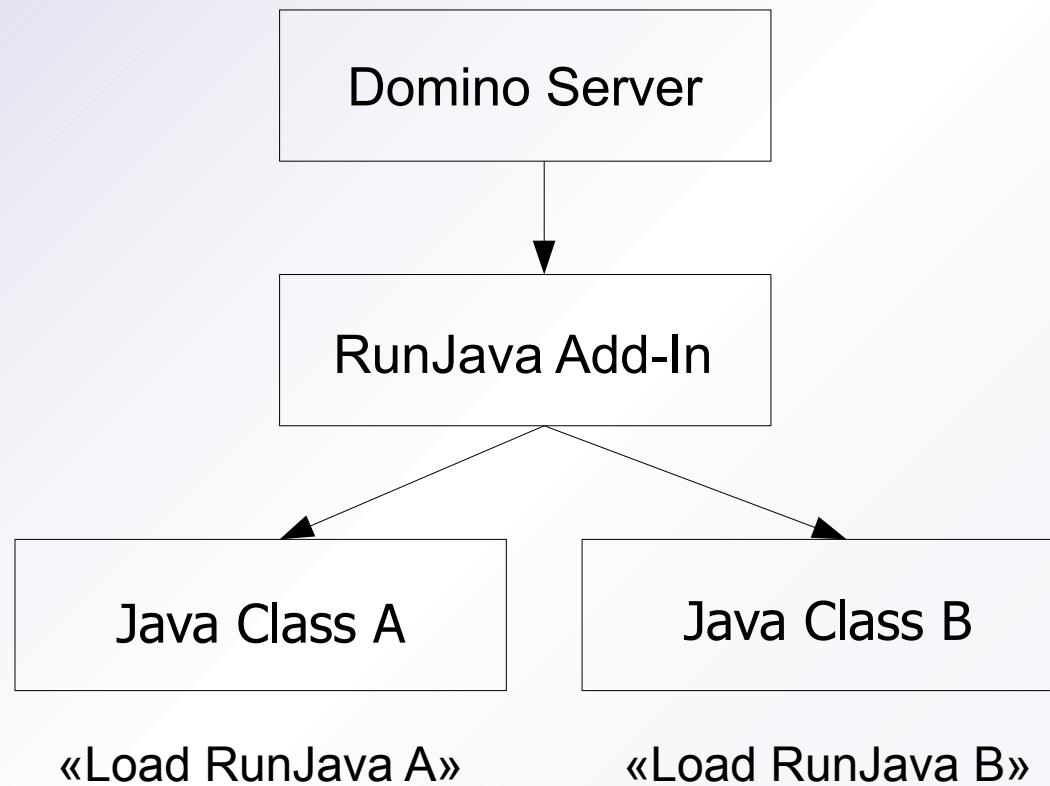
Vorteile von Java unter Domino

- Unabhängig von Domino Plattform (Windows, Linux, OS/400, usw.)
- Unabhängig von Prozessor-Architektur (Win32, Win64, RISC, usw.)
- Moderne Sprache mit guter Entwicklungsumgebung (Eclipse)
- Kann auch ohne Domino Designer entwickelt werden (Lizenz)
- Pure Java, keine Zusätze notwendig (wie z.B. J2EE, SWT, AWT, JSF, usw.)
- Viele frei verfügbare Klassen-Bibliotheken

Java Implementation

- Alle Java Add-Ins laufen unter Kontrolle der Domino Task «RunJava» (wie iSpy, RMEval, usw.)
- Starten einer Java Task:
«Load RunJava *AddInName*» (*AddInName* ist case-sensitive)
- Anzeigen der aktiven Java Tasks
«Tell RunJava Show Tasks» und «Show Tasks»
- Stoppen einer aktiven Java Task
«Tell *AddInName Quit*» oder «Tell RunJava Unload *AddInName*»
- Übergabe von Befehlen an die Java Task wie gewohnt mit
«Tell *AddInName Parameter*»

RunJava Implementation



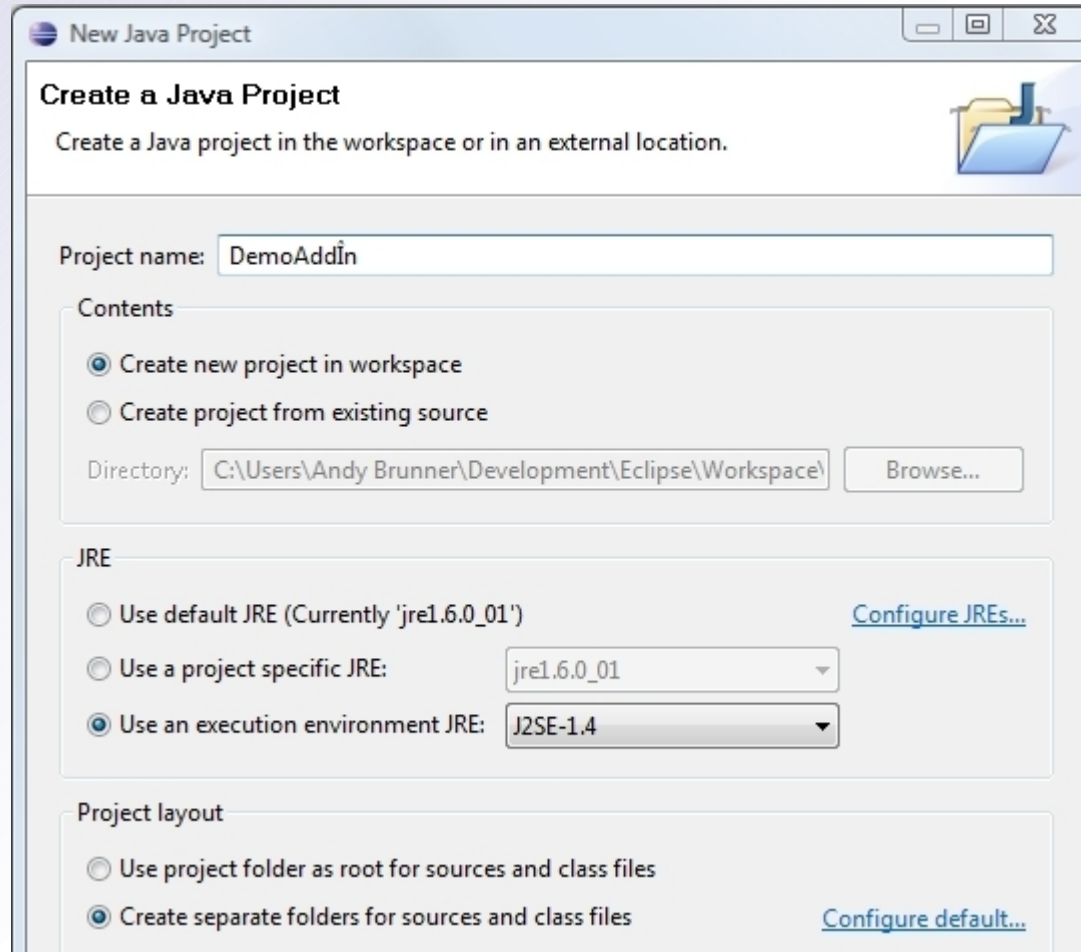
Nachteile von Java unter Domino

- Keine automatische Garbage-Collection von Domino Objekten -> recycle()
- Java Thread Management muss programmiert werden

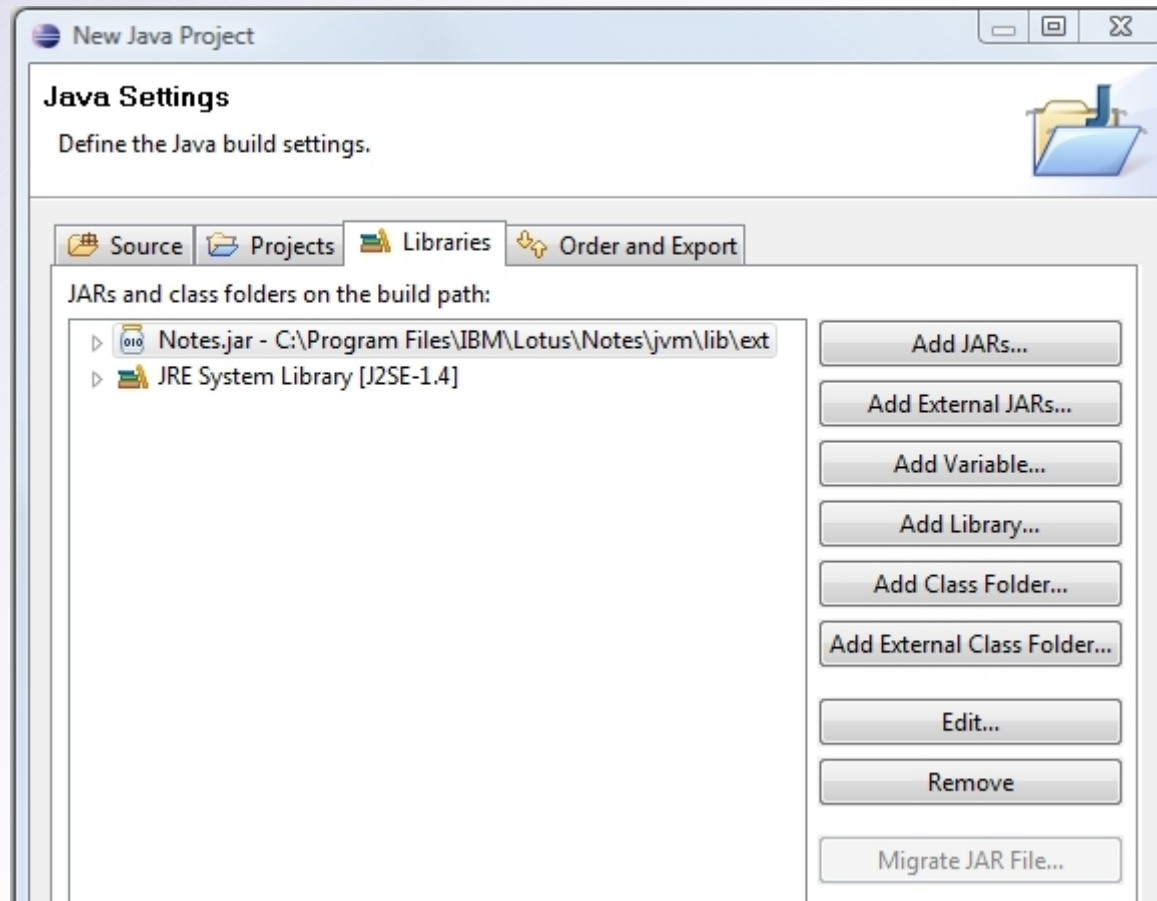
Vorbereitung Entwicklungsumgebung

- Installation der zu unterstützenden Java VM, z.B. Sun JVM 1.4.2
- Installation von Eclipse, z.B. 3.4.1 (Ganymede)
- Erstellen eines Eclipse Projektes für das Add-In
 - Anpassen der zu verwendenden JVM Version
 - Anpassen der JVM Versions-Kompatibilität
 - Notes.jar als externes JAR angeben (Verzeichnis ../domino/jvm/lib/ext)

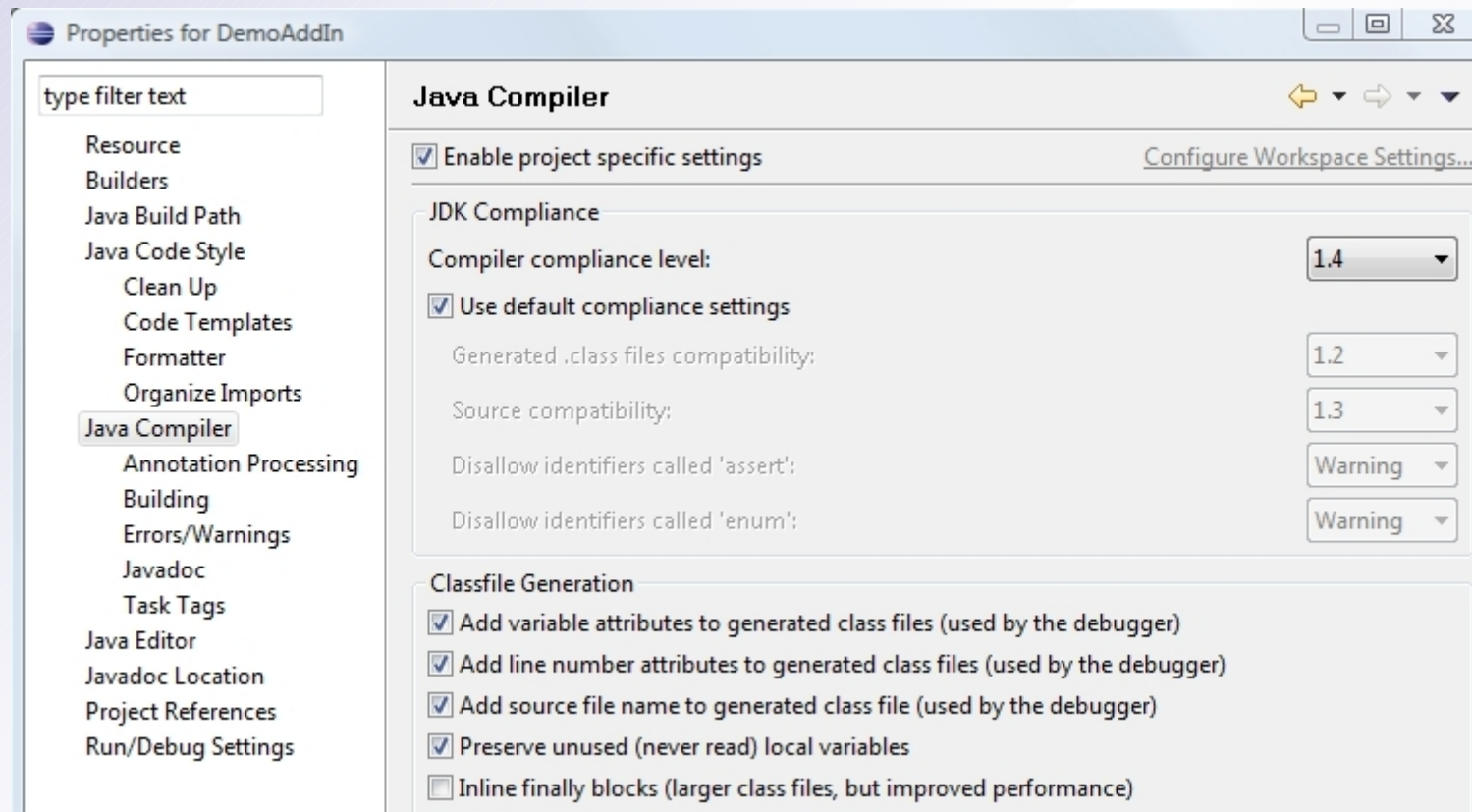
Eclipse Projekt erstellen 1/3



Eclipse Projekt erstellen 2/3



Eclipse Projekt erstellen 3/3



Add-In Installation

- Kopieren aller .class Dateien in das Domino Programm-Verzeichnis
z.B. Win32: ...\\IBM\\Lotus\\Domino
- Starten mit:
«Load RunJava *AddInName*»

Hello World Add-In

```
import lotus.notes.addins.JavaServerAddin;  
  
public class DemoAddin extends JavaServerAddin  
{  
    public void runNotes()  
    {  
        AddInLogMessageText("Hello Java Add-In World !", 0);  
    }  
}
```

Load RunJava DemoAddin

```
03.12.2008 12:45:18 JVM: Java Virtual Machine initialized.  
03.12.2008 12:45:18 RunJava: Started DemoAddin Java task.  
03.12.2008 12:45:18 Hello Java Add-In World !  
03.12.2008 12:45:20 RunJava: Finalized DemoAddin Java task.  
03.12.2008 12:45:21 RunJava shutdown.
```

Demo

Eclipse Projekt «HelloWorld»

Message Queue 1/2

- Alle Verarbeitungen für das Add-In (Befehle, Events) werden über die Message Queue übergeben
- Der Name der Message Queue muss ein spezielles Format aufweisen
MSG_Q_PREFIX + addInName.toUpperCase().replaceAll(" ", "");
- Erstellen, Öffnen und Schliessen der Message Queue:

```
import lotus.notes.internal.MessageQueue;
```

```
MessageQueue messageQueue = new MessageQueue();  
messageQueueState = messageQueue.create(messageQueueName, 0, 0);
```

```
if (messageQueueState == MessageQueue.ERR_DUPLICATE_MQ) {  
    AddInLogMessageText("The Add-In is already running", 0);  
    return;  
}
```

```
messageQueue.open(messageQueueName, 0);  
messageQueue.close(0);
```

Message Queue 2/2

- Nächste Nachricht von der Message Queue lesen oder 3 Sekunden Timeout
`mqState = messageQueue.get(commandLine, 256,
messageQueue.MQ_WAIT_FOR_MSG, 3000);`
- `mqState == MessageQueue.ERR_MQ_QUITTING`
Das Add-In wurde gestoppt oder ein Domino Server Shutdown eingegeben. Die «Quit» und «Exit» Befehle erscheinen nicht als Parameter in der Queue !
- `mqState == MessageQueue.ERR_MQ_TIMEOUT`
Der Timeout ist abgelaufen ohne einen Eintrag in der Message Queue.
Abfragen wie `AddInDayHasElepsed()` sind möglich.

Add-In Task Name

- setName(«Demo AddIn»)
Setzen des sichtbaren Namen des Add-Ins für «Show Tasks» Liste

Add-In Task Status

- `taskID = AddInCreateStatusLine(«Demo AddIn»)`
Erstellt eine Statuszeile für den «Show Tasks» Befehl
- `AddInSetStatusLine(taskID, "Idle")`
Setzt die Statuszeile für den «Show Tasks» Befehl
- `AddInDeleteStatusLine(taskID)`
Löschen der Statuszeile

Domino Konsole Meldungen

- `AddInLogMessageText(«Meldung»)`
Zeigt eine frei formatierbare Meldung auf dem Serverlog an
- `AddinLogErrorText(«Fehlermeldung»)`
Schreibt eine frei formatierbare Fehlermeldung in das Serverlog
- `System.out.println(«Message»)`
Zeigt eine Meldung mit dem Prefix «RunJava JVM: » an (für Testzwecke und Debugging)

Demo

Eclipse Projekt «DemoAddIn»

POP3 Collect Freeware Projekt


Motivation:

- Bedürfnis von mehreren Kunden, POP3 Mails von externen Providern regelmässig in die Domino Mailboxen zu überführen
- Mit Domino-Werkzeugen, statt externe Tools wie POPBeamer, usw.
- Meine erste Servertask (Add-In) unter Domino zu schreiben
- Keine Lust, wieder in C++ zu programmieren
- Einfache Konfiguration der POP3 Server und POP3 Accounts
- Das Projekt als Freeware zu vertreiben

POP3 Collect: Konfigurations-DB

POP3 Collect

Version 0.8.0
Freeware



1. Quickstart
2. Connections
3. Accounts
4. Checkpoints
5. Exit Application

(c) Copyright 2008
ABData, Zürich Switzerland
All Rights Reserved
<http://ABData.CH>

notes net.ch

New Connection Enable Disable

Connection Name	Interval	Schedule	POP3 Host Name
✓ Google Mail	60 min		POP.GoogleMail.Com

POP3 Collect: Quickstart

POP3 Collect Quickstart

Introduction

- Description:
- The Domino addin task POP3Collect reads messages from one or more POP3 servers and sends them to an SMTP server (Domino, Exchange, Sendmail or any other compatible SMTP Server).

Installation

- Updates:
- The current version of this program is always available on <http://NotesNet.CH> (see Downloads/Demos). During startup, this program will also check if a newer version is available. This will be indicated with a message during initialization.
- Prerequisites:
- Lotus Domino 7.0 or higher (with the builtin Java VM)
 - This Domino addin task is entirely written in Java and should therefore run on any Domino 7 (or higher) server platform and processor architecture supported by the Domino JVM.
- Installation steps:
- Create a new database on the target server from the template *POP3Collect.nsf*. The new database can have any name and can be placed on any data directory on the server.
 - Open the newly created database and configure the *Connections* and *Accounts* documents.
 - Detach the following two Java class files and copy them to the servers Domino program directory.



POP3Collect.class



POP3CollectThread.class

- Start the POP3Collect task with the console command or program document:
Load RunJava POP3Collect [Database]
where *Database* is the path and name of the newly created configuration database. The default is *POP3Collect.nsf*. Please note that the Java task name *POP3Collect* is case sensitive.

POP3 Collect: Connections

Google Mail (POP.GoogleMail.Com to Server1.ABData.CH)

Connection

Name: Google Mail
Scheduling interval: 60 Minutes
Scheduling time: No Yes
Logging level: Normal
Enabled: Yes No

POP3 Server

Host name: POP.GoogleMail.Com
Use SSL/TLS: No Yes
Port number: 995

SMTP Server

Host name: Server1.ABData.CH
Use SSL/TLS: No Yes
Port number: 25
Authentication: No Yes

POP3 Collect: Accounts

Google Mail (Anybody@GMail.Com to Andy.Brunner@ABData.CH)

Connection

Use connection: Google Mail

Enabled: Yes No

POP3 Account

User name: Anybody@GMail.Com

Password: *****

Leave mail: No Yes

SMTP Account

Email address: Andy.Brunner@ABData.CH

History

Created: 02.12.2008 08:39:01 Andy Brunner/ABData/CH

Updated: 02.12.2008 08:39:22 Andy Brunner/ABData/CH

Demo

POP3 Collect

POP3 Collect: Server Konsole

> Load RunJava POP3Collect

02.12.2008 08:42:28 JVM: Java Virtual Machine initialized.

02.12.2008 08:42:28 RunJava: Started POP3Collect Java task.

02.12.2008 08:42:28 POP3 Collect: Version 0.8.0 (Freeware) - (c) Copyright 2008 ABData, Zürich Switzerland

02.12.2008 08:42:28 POP3 Collect: Initialization in progress

02.12.2008 08:42:32 POP3 Collect: Configuration successfully loaded

02.12.2008 09:19:06 POP3 Collect: Message delivered to Server1.ABData.CH for Andy.Brunner@ABData.CH from Anybody@GMail.Com Size: 1K

POP3 Collect: Erfahrungen

- Dokumentation der Java Add-In APIs nicht öffentlich verfügbar (Java Implementation aber sehr nahe an entsprechenden C++ Add-In APIs)
- Sehr wenige Code-Beispiele, z.B:
<http://www.nsftools.com/tips/JavaAddinTest.java> (Julian Robichaux)
- Festlegung auf unterstützte Domino JVM 1.4.2+ (d.h. Domino 7.0+)
- JavaMaxHeapSize mit 64MB zu knapp (kann mit Notes.Ini Eintrag vergrößert werden, z.B. JavaMaxHeapSize=256MB)
- Keine Probleme mit Java unter Domino
- Über 100 Server-Installationen im Einsatz (Tests, Migrations-Szenarien und produktive Server)
- Installierte Plattformen (Win32, Win64, OS/400, Unix)

Tips & Hints

- Die Ressourcen (z.B. Memory) aller Domino Objekte müssen mit der Methode `recycle()` freigegeben werden
- Vergrössern des Java Heap Space mit der Notes.Ini Variable `JavaMaxHeapSize=xxxMB` (Default ist 64MB, empfohlen 256MB)
- Default Java Package Namen verwenden, damit keine langen Add-In Namen entstehen, wie «Load RunJava *com.firma.package.AddInName*»
- Längere Verarbeitungen in einer Subtask durchführen, damit die Message Queue nicht blockiert wird
- Alle Domino Ressourcen (z.B. MessageQueue) müssen geschlossen werden (z.B. über `finalize()` Methode)
- Die Java .class Dateien bleiben im Cache. Um eine neue Version zu laden, muss RunJava gestoppt werden

Links

- NotesNet.CH Vereinigung
<http://www.notesnet.ch>
- POP3 Collect Freeware
<http://notesnet.ch/notesnet/notesnet.nsf/web/downloads>
- Eclipse Downloads, z.B. Eclipse Classic 3.4.1 (Ganymede)
<http://www.eclipse.org/downloads/>
- Java J2SE Downloads, z.B. V1.4.2 (End of Service Life)
<http://java.sun.com/j2se/1.4.2/download.html>

Links: Dokumentation

- Lotus C API Notes/Domino 7.0 Reference:
<http://www-12.lotus.com/ldd/doc/tools/c/7.0/api70ref.nsf>
- Lotus C API Notes/Domino 7.0 User Guide:
<http://www-12.lotus.com/ldd/doc/tools/c/7.0/api70ug.nsf>
- Sample Code von Julian Robichaux:
<http://www.nsftools.com/tips/JavaAddinTest.java>

Entwicklung einer Java Add-In Servertask



Vielen Dank für Ihre Aufmerksamkeit !

Fragen ?

Andy.Brunner@ABData.CH