NOTES & DOMINO ENTWICKLERCAMP

assono

EntwicklerCamp 2013
Track 3, Session 2:

Reite den Mustang
Java-Entwicklung in Notes & Domino

Gelsenkirchen, 11. März 2013

Innovative Software-Lösungen.

www.assono.de

Thomas Bahn

Diplom-Mathematiker

Java und relationale Datenbanken seit 1997

Notes & Domino seit 1999

regelmäßig Sprecher auf Fachkonferenzen

mehrfacher Fachartikel-Autor (THE VIEW)

tbahn@assono.de
http://www.assono.de/blog
04307/900-401

assono
IT-Consulting & Solutions

Agenda

- Einführung in die Programmiersprache Java
- Java in Lotus Notes & Domino
  - Applets
  - Agenten
  - Servlets
  - (Standalone-)Anwendungen
  - XPages
  - Plug-ins
  - der Rest...

Agenda

- Einführung in die Programmiersprache Java
- Java in Lotus Notes & Domino
  - Applets
  - Agenten
  - Servlets
  - (Standalone-)Anwendungen
  - XPages
  - Plug-ins
  - der Rest...

Die Programmiersprache Java

- Syntax ähnlich wie C, C++ und JavaScript
- objektorientiert
- kompiliert in Zwischensprache, den Byte-Code
- Byte-Code wird in Java Virtual Maschine (JVM) ausgeführt (interpretiert) – vergleichbar LotusScript
- Just-in-Time-Compiler sorgt für richtige Performance

Size matters

Java ist **case-sensitive**.

Überall! Immer!

Anweisungen

- Anweisungen in Java werden **mit Semikolon – `;` – abgeschlossen**
- ähnlich wie in Formelsprache, aber dort werden sie mit Semikolon **getrennt**
- Beispiel:

```
int x = 4;
```

Anweisungsblöcke

- Überall wo einzelne Anweisungen stehen können, dürfen auch mehrere Anweisungen stehen, die von geschweiften Klammern `{}` zu einem Block zusammengefasst werden.
- Verwendung in Schleifen, bedingten Anweisungen usw.
- Beispiel:

```
{
    int x = 4;
    int y = 5;
}
```

Kommentare

- Kommentar bis zum Zeilenende: `//`
- mehrzeiliger Kommentar: `/* … */`
- Beispiele:

```
int i = 4; // Zuweisung

/*
 * und als nächstes eine Addition...
 */
i = i + 2;
```

Primitive Datentypen

- Ganzzahlen:
  - `byte` (8 bit)
  - `short` (16 bit)
  - `int` (32 bit)
  - `long` (64 bit)
- Gleitkommazahlen:
  - `float` (32 bit)
  - `double` (64 bit)
- Zeichen: `char` (16 bit!)
- Boolean: `boolean` (1 bit)

Datentypen

- **alles andere** sind Objekte, z. B.
  - `java.lang.String`
  - `java.util.Date`
  - `java.lang.Object` (Basis aller Klassen)
  - `java.lang.Class` (auch Klassen sind Objekte!)
- für jeden primitiven Datentyp gibt es auch eine Klasse (großer Anfangsbuchstabe), z. B.
  - `java.lang.Integer`
  - `java.lang.Char`

Variablendeklaration

- Deklaration von Variablen:

```
int i;
String s;
java.util.Date today;
```

- sofortige Initialisierung möglich:

```
int i = 0;
String name = "Thomas Bahn";
java.util.Date now = new java.util.Date();
```

- Konstanten mit `final`-Modifizierer:

```
final int MAX = 100;
final String GREETING = "Hallo Welt";
```

Operatoren

- die üblichen Verdächtigen: `+ - * /` usw.
- Erhöhen/Erniedrigen: `++` und `--`
  vor oder nach dem Term, um vor Auswertung bzw. danach anzuwenden, z. B.:

```
int i = 4;
int j = ++i   // => j == 5, i == 5
int k = i++   // => k == 5, i == 6
```

- Logische Operatoren: `&&` (and), `||` (or), `!` (not)

Vergleichsoperatoren

- Vergleichsoperatoren: ==, <=, >=, !=
- **Vorsicht**:
  Vergleichen mit **doppeltem** Gleichheitszeichen,
  sonst ist es eine **Zuweisung**!

Zuweisungen

- Zuweisen mit =
- Kurzschreibweisen
  - += addieren/anhängen und zuweisen
  - -= subtrahieren und zuweisen
  - usw.
- Beispiele:

```
int i = 4;
i += 2;          // => i == 6
i -= 3;          // => i == 3
String s = "abcd";
s += "efg";   // => s == "abcdefg"
```

Strings

- Zusammenfügen von Strings: `+`
- Strings sind Objekte, also kann man auch ihre Methoden aufrufen, z. B.:

```
"Hallo EntwicklerCamp!".substring(0,4)
                                    // => "Hallo"
```

Sonderzeichen

- spezielle Zeichen mit \ maskiert:
  - \n new line
  - \t horizontal tab
  - \r carrige return
  - \' single quote
  - \" double quote
  - \\ backslash

Schleifen

- `for`-Schleife:

  `for` (*Start*, *Bedingung*, *Erhöhung*) *Anweisung(sblock)*;

- Beispiel:

```
for (int i = 0; i < 5; ++i) {
    System.out.println(i);
}
```

Schleifen (forts.)

- `do`-Schleife (nicht abweisend, also mindestens eine Ausführung):

  `do` *Anweisung(sblock)* `while` (*Bedingung*);

- Beispiel:

```
int i = 0;
do {
    System.out.println(i++);
} while (i < 5);
```

Schleifen (forts.)

- `while`-Schleife (abweisend, also vielleicht keine Ausführung):

  `while` (*Bedingung*) *Anweisung(sblock)*

- Beispiel:

```
int i = 0;
while (i < 5) {
    System.out.println(i++);
}
```

Bedingte Ausführung – `if`

- `if – then – else`:

  `if` (*Bedingung*) *Anweisung(sblock)*
  `else` *Anweisung(sblock)*

- `else if` ist kein "Spezialfall" wie in LotusScript

Bedingte Ausführung – `if` (forts.)

- Beispiel:

```
int i = 0;
if (i < 5) {
    System.out.println(i);
} else if (i < 0) {
    System.out.println("negativ");
} else {
    System.out.println("zu groß");
}
```

Bedingte Ausführung – `switch`

- `switch – case:`

```
switch (primitiver Wert) {
    case (Vergleichswert):
    Anweisungen
    break;

    case (weiterer Vergleichswert):
    Anweisungen
    break;

    default:
    Anweisungen
}
```

Bedingte Ausführung – `switch` (forts.)

- Beispiel:

```
switch (i) {
    case (0):
    System.out.println("kein");
    break;

    case (1):
    System.out.println("ein");
    break;

    default:
    System.out.println(i);
}
```

- ohne `break` wird mit dem nächsten `case` weiter gemacht (und `default`!)

Arrays

- ähnlich wie in LotusScript
- Es gibt Array-Konstanten, Schreibweise:

```
int[] anArray = {100, 200, 300, 400, 500};
anArray[0] = 110;
System.out.println(
    "Element 3 at index 2: " + anArray[2]
); // => 300
```

- erstes Element hat den Index **0**!
- Array.`length` gibt die Länge zurück

Listen

- Mehrere Klassen für Listen:
  - `java.util.List<E>` (Interface, keine Klasse!)
  - `java.util.Vector<E>`
  - `java.util.ArrayList<E>`
  - `java.util.LinkedList<E>`
  - usw.
- Listen können "typisiert" werden (ab Java 5), z. B.

```
java.util.ArrayList<String>
```

- Durchlaufen mit `for`-Schleife oder `Iterator`

Klassen

- ähnlich wie in LotusScript, aber viel mächtiger
- Beispiel:

```
class Rectangle extends Shape {
    // Variablen, Methoden, …
}
```

- "Hauptprogramm" ist `main`-Methode einer Klasse:

```
public static void main(String[] args) { … }
```

- in Java programmiert man ausschließlich mit Klassen!

Modifizierer

- Es gibt viele Modifizierer für Klassen, Variablen, Methoden...
  - `public` öffentlich, also für alle sichtbar
  - `protected` nur eigenes Package
  - `private` nur Klasse und Unterklassen
  - `static` gehört zur Klasse, nicht zum Objekt
  - `final` unveränderlich = Konstante
  - `abstract` wird erst in Unterklasse definiert
  - usw.

Interfaces

- Interfaces (Schnittstellen):
  ähnlich wie Klassen, aber ohne „Fleisch“
- „Verträge“ die definieren, **was** es gibt, **nicht wie** etwas gemacht wird
- eine Klasse kann mehrere Interfaces implementieren, aber nur von einer Erben
- Variablen dürfen Interface als Typ haben
- sehr wichtig in Java, um Klassen zu "entkoppeln", damit man die Implementierung austauschen kann

Interfaces (forts.)

- Beispiel:

```
interface Area {
    public double calcArea();
}

class Rectangle implements Area {
    public double calcArea() {…}
    // weitere Variablen, Methoden, …
}

class Circle implements Area {…}

Area a = new Rectangle(…);
System.out.println("Fläche: " + a.calcArea());
a = new Circle(…);
System.out.println("Fläche: " + a.calcArea());
```

Packages und Imports

- Mehrere Klassen können in Paketen (Packages) zusammen gefasst werden, z. B. java.util.
- Kurzschreibweise:
  Damit man nicht jedes Mal den "vollen" Namen schreiben muss, kann man Pakete mit `import` importieren.
- steht am Anfang der Java-Datei
- Einzelimport oder mit *-Wildcard

Packages und Imports (forts.)

- Beispiel: statt

```
java.util.List liste = new java.util.ArrayList();
java.util.ListIterator it =
                liste.listIterator(liste.size());
```

- geht auch:

```
import java.util.List;
import java.util.ArrayList;
import java.util.ListIterator;

List liste = new ArrayList();
ListIterator it =
                liste.listIterator(liste.size());
```

Packages und Imports (forts.)

- oder noch kürzer:

```
import java.util.*;

List liste = new ArrayList();
ListIterator it =
             liste.listIterator(liste.size());
```

Mehr zu Klassen

- **Konstruktor** hat gleichen Namen wie die Klasse
- **Destruktor** heißt `finalize` (LotusScript: `Delete`)
- Beispiel:

```
class Shape {
    public void Shape() { ... } // Konstruktor
    public void finalize() { ... } // Destruktor
}
```

- `finalize` wird "irgendwann" vom Garbage Collector aufgerufen, nachdem das Objekt nicht mehr benutzt werden kann, also die letzte Referenz gelöscht wurde (wenn überhaupt)

Typumwandlung

- zwischen primitiven Datentypen und "ihren" Objekttypen automatisch (Autoboxing) seit Java 5
- zwischen Objekten (Klassen und Unterklassen) nur "manuell" per Casting
- Beispiel:

```
ArrayList liste = ...;
String element = (String) liste.get(i);
// liste.get gibt Wert mit Typ Object zurück
```

Typumwandlung (forts.)

- Casten geht nur vom Allgemeinen zum Speziellen, also von der Oberklasse zur Unterklasse:
  - ein `String` ist sicher ein `Object`
  - ein `Object` muss kein `String` sein
- Compiler kann Cast nicht prüfen:
  Gefahr von Laufzeitfehlern

Überladen

- Anders als in LotusScript kann eine Klasse **mehrere Methoden mit gleichem Namen** haben – solange sich die Parameterliste (Signatur) unterscheidet.
- Beim Aufruf wird "passende" Methode gesucht.
- auch Konstruktoren können überladen werden

Überladen (forts.)

- Beispiel:

```
class Ellipse extends Shape {
   public Ellipse(Point center, float radiusX,
                  float radius_y) {...};
   public Ellipse(Point center_1,
                  Point center_2) {...};
}
...
Ellipse e =
        new Ellipse(new Point(1,2), 2.0, 4.0);
```

Ein- und Ausgabe

- nutzbar, nur wenn es eine Konsole gibt:
  - System.out Ausgabe-Stream
  - System.err Fehlerausgabe-Stream
  - System.in Eingabe-Stream
- Beispiel:

```
do {
    System.out.println(new java.util.Date());
    Thread.sleep(1000);
} while (System.in.available() == 0);
```

Fehlerbehandlung

- Exceptions (Ausnahmen) und Errors (Fehler)
- `try` *Anweisung(sblock)*
  `catch` (*XException xe*) *Anweisung(sblock)*
  `catch` (*YException ye*) *Anweisung(sblock)*
  …
  `finally` *Anweisung(sblock)*
- statt On Error... in LotusScript
- ähnlich wie in JavaScript, aber mehrere catches möglich (da unterschiedliche Exception-Klassen)

Agenda

- Einführung in die Programmiersprache Java
- Java in Lotus Notes & Domino
  - Applets
  - Agenten
  - Servlets
  - (Standalone-)Anwendungen
  - XPages
  - Plug-Ins
  - der Rest...

Applets

- jupps, die gibt’s immer noch
- laufen im Browser und im Notes-Client
- aus Sicherheitsgründen viele Beschränkungen

Applets (forts.)

Demo: Hello World-Applet

Hello World-Applet (Ausschnitt)

```
public class HelloWorldApplet extends javax.swing.JApplet {
   public void init() {
      getContentPane().add(new ContentPane());
   }
   class ContentPane extends javax.swing.JPanel {
      javax.swing.JLabel label;
      public ContentPane() {
         setLayout(null);
         label = new javax.swing.JLabel("Hallo EntwicklerCamp!");
         label.setBounds(10, 10, 200, 20);
         label.setBackground(java.awt.Color.WHITE);
         label.setOpaque(true);
         label.setHorizontalAlignment(SwingConstants.CENTER);
         add(label);
      }
      public void paintComponent(java.awt.Graphics g) {
         super.paintComponent(g);
         setBackground(java.awt.Color.WHITE);
         setForeground(new java.awt.Color(207, 37, 57));
         g.fillRect(8, 8, 204, 24);
      }
   }
}
```

Java-Konsole im Notes-Client

- zum Öffnen der Java-Konsole im Notes-Client:
  Tools-Menü → Show Java Debug Console
- zeigt die System.out von Applets und Agents
- hilfreich beim „Debuggen“ (the old way)

Applets (forts.)

Demo: Warnton-Applet

Warnton-Applet (Ausschnitt)

```
public class WarntonApplet extends AppletBase {

AudioClip clip = null;
Session session = null;
Database db = null;
View monitoredView = null;

public void notesAppletInit() {
   clip = getAudioClip(getCodeBase(), "extreme_alarm.wav");
   try {
      NotesThread.sinitThread();
      session = getSession();
      db = getContext(session).getDatabase();
      monitoredView = db.getView("MonitoringView");
   } catch (NotesException ne) {…}
}

public void notesAppletStart() {
   monitorActive = true;
   monitorView();
}
```

Warnton-Applet (Ausschnitt, forts.)

```
void monitorView() {
   try {
      do {
         monitoredView.refresh();
         int entriesInView = monitoredView.getEntryCount();

         if (entriesInView > 0) {
            clip.play();
         } else if (entriesInView == 0) {
            clip.stop();
         }

         try { // warte ein paar Sekunden
            Thread.sleep(WAIT_TIME);
         } catch (InterruptedException ie) { } // do nothing
      } while (monitorActive);
   } catch (NotesException ne) {…}
}

public void notesAppletStop() {
   clip.stop();
   monitorActive = false;
}
```

Warnton-Applet (Ausschnitt, forts.)

```
public void notesAppletDestroy() {
   try {
      if (monitoredView != null) {
         monitoredView.recycle();
      }
      if (db != null) {
         db.recycle();
      }
      if (session != null) {
         closeSession(session);
      }
   } catch (NotesException ne) {
      System.out.println("Notes-Fehler #" + ne.id + ": " + ne.text);
   } catch (Exception e) {
      e.printStackTrace();
   } finally {
      NotesThread.stermThread();
   }
}
```

Warnton-Applet (forts.)

- **Recycling ist erste (Java-)Bürgerpflicht!**
- Für jedes Java-Objekt (session, database, view, document usw.) gibt es im Hintergrund ein C-Objekt, dass ohne `recycle()` nie mehr frei gegeben wird.
- Das **Nummer 1-Problem** für Neulinge.
- Aber auch nicht zu früh freigeben (in nebenläufigen Java-Programmen)...
- verwendeter Warnton:
  extreme_alarm.wav von sirplus
  www.freesound.org/samplesViewSingle.php?id=25032

Agenda

- Einführung in die Programmiersprache Java
- Java in Lotus Notes & Domino
  - Applets
  - Agenten
  - Servlets
  - (Standalone-)Anwendungen
  - XPages
  - Plug-Ins
  - der Rest...

Agenten

- ähnlich wie LotusScript-Agenten
- kein Zugriff auf das Frontend (geöffnetes Dokument, Ansicht, aktuelle Auswahl usw.)
- Vorteil: können viele vorhandene Java-Bibliotheken nutzen
- Haupteinsatzgebiete:
  - geplante Agenten, die eine Schnittstelle zu anderen Systemen implementieren, z. B. zum Datenabgleich
  - WebQueryOpen/WebQuerySave

Agenten (forts.)

Demo: Hello World-Agent

Hello World-Agent (Ausschnitt)

```
public class JavaAgent extends AgentBase {

    public void NotesMain() {
      try {
          Session session = getSession();
          AgentContext agentContext = session.getAgentContext();
          Agent currentAgent = agentContext.getCurrentAgent();

          System.out.println("Hallo EntwicklerCamp!");
          System.out.println("Der Agent \"" +
                  currentAgent.getName() + "\" wurde von " +
                  session.getCommonUserName() + " gestartet.");

          currentAgent.recycle();
          agentContext.recycle();
          session.recycle();
      } catch(Exception e) {
          e.printStackTrace();
      }
    }
}
```

Agenten (forts.)

Demo: PDF-Export-Agent

PDF-Export-Agent (Ausschnitt)

```
com.itextpdf.text.Document document =
                                        new com.itextpdf.text.Document();
try {
   PdfWriter.getInstance(document, new
                                   FileOutputStream("AllEntries.pdf"));
   document.open();
   float[] widths = { 0.35f, 0.65f };
   PdfPTable table = new PdfPTable(widths);

   PdfPCell cell = new PdfPCell(new Paragraph("Aktuelle Einträge",
           new Font(Font.FontFamily.HELVETICA,16,Font.BOLD)));
   cell.setColspan(2);
   table.addCell(cell);

   cell = new PdfPCell(new Paragraph("Priorität",
           new Font(Font.FontFamily.HELVETICA, 12, Font.BOLD)));
   table.addCell(cell);

   cell = new PdfPCell(new Paragraph("Thema",
           new Font(Font.FontFamily.HELVETICA, 12, Font.BOLD)));
   table.addCell(cell);
```

PDF-Export-Agent (Ausschnitt)

```
Document doc = allEntriesView.getFirstDocument();
Document nextDoc = null;
while (doc != null) {
   nextDoc = allEntriesView.getNextDocument(doc);
   switch (Integer.parseInt(doc.getItemValueString("Priorität"))) {
   case 4:
      table.addCell("4 - super wichtig");
      break;
   case 3:
      table.addCell("3 - noch wichtiger");
      break;
   case 2:
      table.addCell("2 - extrem super wichtig");
      break;
   case 1:
      table.addCell("1 - unvorstellbar wichtig");
   }
   table.addCell(doc.getItemValueString("Thema"));

   doc.recycle();
   doc = nextDoc;
}
```

PDF-Export-Agent (Ausschnitt)

```
   document.add(table);

} catch (DocumentException de) {
   System.err.println(de.getMessage());
} catch (IOException ioe) {
   System.err.println(ioe.getMessage());
}

document.close();
```

iText

- iText ist eine Open Source-Bibliothek (GNU Affero General Public License v3) zum Erstellen und Bearbeiten von PDF-Dateien
- Web-Seite: http://itextpdf.com
- IBM developerWorks: http://www.ibm.com/developerworks/opensource/library/os-javapdf/

Agenda

- Einführung in die Programmiersprache Java
- Java in Lotus Notes & Domino
  - Applets
  - Agenten
  - Servlets
  - (Standalone-)Anwendungen
  - XPages
  - Plug-Ins
  - der Rest...

Servlets

- kommen ursprünglich von Java-Web-Servern
- auch der Domino-Server ist ein "Servlet-Container"
- viele Ähnlichkeiten mit Agenten
- werden normalerweise per URL aufgerufen
- erzeugen häufig Web-Seite als Ergebnis
- Vorteil gegenüber Agenten:
  - werden nur einmal initiiert, ab dem zweiten Aufruf viel schneller

Servlets (forts.)

Demo: Hello World-Servlet

Hello World-Servlet (Ausschnitt)

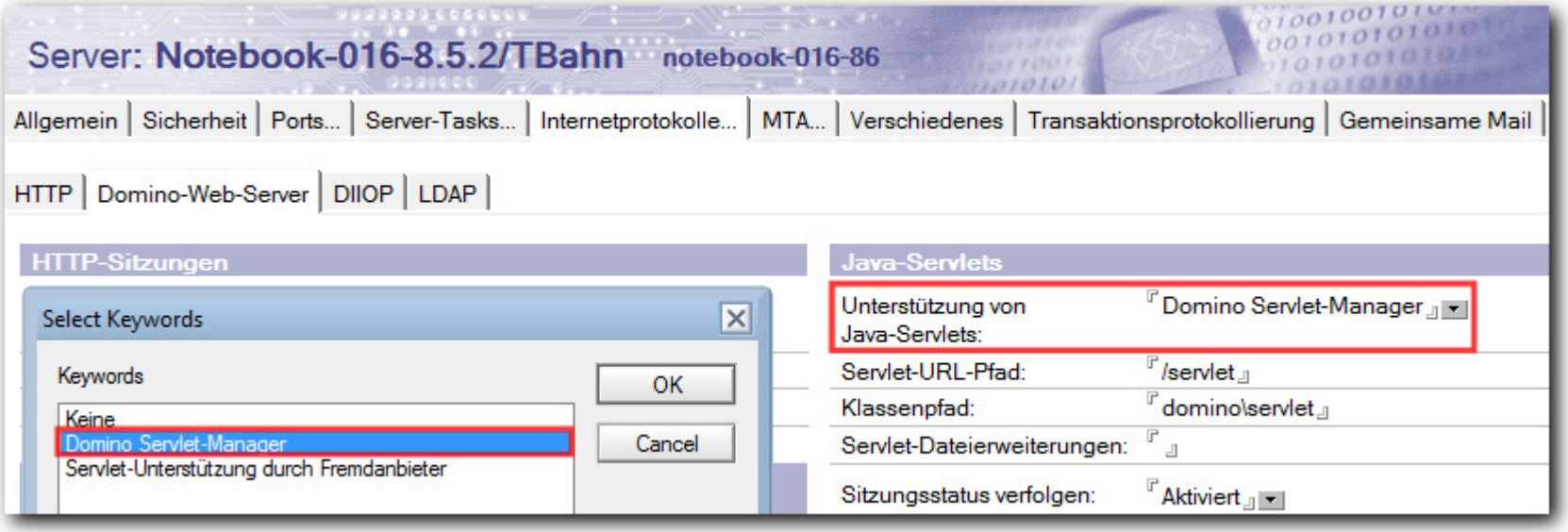
```
public class HelloWorldServlet extends HttpServlet {
   Session session = null;
   public void init(ServletConfig config) throws ServletException {
      try {
         NotesThread.sinitThread();
         session = NotesFactory.createSession();
      } catch (NotesException ne) {
         System.out.println("Fehler #" + ne.id + ": " + ne.text);
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
   public void destroy() {
      try {
         session.recycle();
         NotesThread.stermThread();
      } catch (NotesException ne) {
         System.out.println("Fehler #" + ne.id + ": " + ne.text);
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
```

Hello World-Servlet (Ausschnitt, forts.)

```
public class HelloWorldServlet extends HttpServlet {
   ...
   protected void doGet(HttpServletRequest request, HttpServlet-
         Response response) throws ServletException, IOException {
      super.doGet(request, response);
      response.setContentType("text/plain");
      PrintWriter writer = response.getWriter();
      writer.println("Hallo EntwicklerCamp!\n");
      try {
         writer.println("Serverstatus: " + session.
            sendConsoleCommand(session.getServerName(),
            "show server") + "\n");
         writer.println("Aktuelle Tasks: " + session.
            sendConsoleCommand(session.getServerName(),
            "show tasks only") + "\n");
      } catch (NotesException ne) {
         System.out.println("Fehler #" + ne.id + ": " + ne.text);
      } catch (Exception e) {
         e.printStackTrace();
      }
      writer.close();
   }
}
```

Servlet-Engine aktivieren

- Servlet-Engine des Domino-Servers per Vorgabe deaktiviert, anschalten im Server-Dokument:



- Servlet-URL-Pfad: wenn URL die folgende Form hat:
  http(s)://server**/servlet**/ServletName
  wird das Servlet „ServletName“ gestartet

Servlets konfigurieren

- im Daten-Verzeichnis des Domino-Servers die Datei servlets.properties erstellen und Servlets eintragen, Beispiel:

```
servlet.HelloWorld.code=de.assono.HelloWorldServlet.class
servlet.DominoHelper.code=de.assono.DominoHelperServlet.class
servlets.startup = DominoHelper
```

- `servlets.startup`: diese Servlets werden beim Start des HTTP-Tasks gleich mit gestartet und initialisiert
- Man kann auch Aliase und Parameter definieren
- URL zum Aufruf des Servlets dann z. B. http://notebook-019-9/servlet/HelloWorld

Servlets (forts.)

Demo: Domino-Helper-Servlet

Domino-Helper-Servlet (Ausschnitt)

```
private String getQotD() {
   final String QOTD_FEED_URL =
                  "http://feeds.feedburner.com/quotationspage/qotd";
   String quote = "";
   RSSHandler handler = new RSSHandler();
   try {
      RSSParser.parseXmlFile(new URL(QOTD_FEED_URL),handler,false);
      RSSChannel channel = handler.getRSSChannel();
      LinkedList list = channel.getItems();
      int randomPos = (int) Math.floor(Math.random()*list.size());
      RSSItem itm = (RSSItem) list.get(randomPos);
      String desc = itm.getDescription();
      quote = desc.substring(0, (desc + "<p>").indexOf("<p>")) +
                                          ", <i>" + itm.getTitle() +
"</i>\n";
   } catch (MalformedURLException e) {
      e.printStackTrace();
   } catch (RSSException e) {
      e.printStackTrace();
   }
   return quote;
}
```

RSSLib for J

- RSSLib for J: Open Source-Bibliothek (GNU General Public License) zum Parsen und Extrahieren von Informationen aus RSS- und Atom-Feeds
- Web-Seite: http://sourceforge.net/projects/rsslib4j/
- Verwendeter Feed von The Quotations Page: http://www.quotationspage.com/

Agenda

- Einführung in die Programmiersprache Java
- Java in Lotus Notes & Domino
  - Applets
  - Agenten
  - Servlets
  - (Standalone-)Anwendungen
  - XPages
  - Plug-Ins
  - der Rest...

(Standalone-)Anwendungen

- normale Java-Anwendungen mit oder ohne grafische Benutzeroberfläche, die "auch mit Notes/Domino kommunizieren"
- maximale Flexibilität
- viel Funktionalität über Open Source-Bibliotheken kostenlos verfügbar und leicht nutzbar

(Standalone-)Anwendungen (forts.)

Demo: Hello World-App

Hello World-App (Ausschnitt)

```
public class HelloWorldApp {
   public static void main(String[] args) {
      final String DB_FILE_PATH = "EC11-T4S4-Java in Notes.nsf";
      System.out.println("Hallo EntwicklerCamp!\n");
      try {
         Session s = NotesFactory.createSession("Server");
         if (s != null) {
            Database db = s.getDatabase("", DB_FILE_PATH);
            System.out.println("Benutzer: "+ s.getCommonUserName());
            System.out.println("Datenbank: "+ db.getTitle());
            System.out.println("Anzahl Dokumente in der Datenbank: "
                               + db.getAllDocuments().getCount());
            db.recycle();
            s.recycle();
         }
      } catch (NotesException ne) {
         System.out.println("Fehler #" + ne.id + ": " + ne.text);
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

(Standalone-)Anwendungen (forts.)

Demo: Importer-App

Importer-App (Ausschnitt)

```
public class ImporterApp {
   public static void main(String[] args) {
      NotesThread.sinitThread();
      try {
         do {
            if (session == null || !session.isValid()) {
               session = NotesFactory.createSession("");
            }
            monitorDirectory();
            Thread.sleep(MONITORING_FREQUENCY);
         } while (System.in.available() == 0);
         session.recycle();
      } catch (NotesException ne) {
         System.out.println("Notes-Fehler #"+ne.id+": "+ne.text);
      } catch (Exception e) {
         e.printStackTrace();
      } finally {
         NotesThread.stermThread();
      }
   }
}
```

Importer-App (Ausschnitt, forts.)

```
private static void monitorDirectory() throws NotesException {

   File importDirectory = new File(IMPORT_DIRECTORY);
   File[] fileList = importDirectory.listFiles();
   for (int i = 0; i < fileList.length; i++) {
      if (fileList[i].isFile() &&
                        fileList[i].getName().endsWith(".dxl")){
         importDXL(fileList[i]);
      }
   }
   if (fileList.length > 0) {
      zipImportedFiles(fileList);
   }
}
```

Importer-App (Ausschnitt, forts.)

```
private static void importDXL(File file) throws NotesException {

   String fileName = file.getName();
   Database db = session.getDatabase("", DB_FILE_PATH);
   Stream stream = session.createStream();
   try {
      if (stream.open(file.getCanonicalPath()) &
           stream.getBytes() > 0) {
         DxlImporter importer = session.createDxlImporter();
         importer.setDocumentImportOption(
                        DxlImporter.DXLIMPORTOPTION_CREATE);
         importer.importDxl(stream, db);
         stream.close();
      }
   } catch (IOException ioe) {
      ioe.printStackTrace();
   }
}
```

Importer-App (Ausschnitt, forts.)

```
private static void zipImportedFiles(File[] importedFiles) {
   df = new SimpleDateFormat("yyyy-MM-dd-HH-mm-ss.'zip'");
   String zipfile = IMPORTED_DIRECTORY + df.format(new Date());
   int bytes_read;
   try {
      ZipOutputStream out = new ZipOutputStream(new
                                          FileOutputStream(zipfile));
      for (int i = 0; i < importedFiles.length; ++i) {
         File file = importedFiles[i];
         FileInputStream in = new FileInputStream(file);
         ZipEntry entry = new ZipEntry(file.getName());
         out.putNextEntry(entry);
         while ((bytes_read = in.read(buffer)) != -1) {
            out.write(buffer, 0, bytes_read);
         }
         in.close();
         if (file.delete()) {
            System.out.println("'" + file.getName() + "' gelöscht");
         }
      }
      out.close();
   } catch (…) {…}
}
```

Agenda

- Einführung in die Programmiersprache Java
- Java in Lotus Notes & Domino
  - Applets
  - Agenten
  - Servlets
  - (Standalone-)Anwendungen
  - XPages
  - Plug-Ins
  - der Rest...

Agenda

- Einführung in die Programmiersprache Java
- Java in Lotus Notes & Domino
  - Applets
  - Agenten
  - Servlets
  - (Standalone-)Anwendungen
  - XPages
  - Plug-Ins
  - der Rest...

XPages

- eigentlich ist JavaScript „die“ Sprache der XPages
- im Hintergrund wird Server-Side JavaScript (SSJS) **in Java** übersetzt
- mit Java-Kenntnissen kann man die Abläufe besser verstehen und nachvollziehen
- Java lässt sich sehr einfach aus SSJS benutzen
- Details im Zusatzmaterial
- Track 1, Session 5: **„XPages und Java“**,
  Bernd Hort, assono
  morgen, Dienstag, 14 Uhr (nach dem Mittagessen)

Agenda

- Einführung in die Programmiersprache Java
- Java in Lotus Notes & Domino
  - Applets
  - Agenten
  - Servlets
  - (Standalone-)Anwendungen
  - XPages
  - Plug-Ins
  - der Rest...

Plug-Ins

- seit Version 8 basiert Notes Standard-Client auf Eclipse Rich Client Platform (RCP) und Lotus Expeditor
- Client ist im Wesentlichen eine "Sammlung" von Plug-Ins
- Erweiterungen über eigene Plug-Ins leicht möglich
- Details im Zusatzmaterial

Agenda

- Einführung in die Programmiersprache Java
- Java in Lotus Notes & Domino
  - Applets
  - Agenten
  - Servlets
  - (Standalone-)Anwendungen
  - XPages
  - Plug-Ins
  - der Rest...

der Rest…

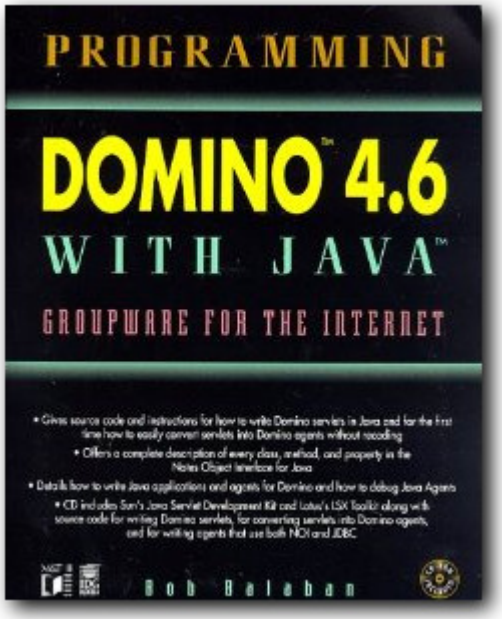
- Web-Service-Anbieter
- Web-Service-Konsumenten
- LS2J (LotusScript to Java-Bridge)
- Domino-Server-Tasks mit JAVAADDIN
  http://www.openntf.org/internal/home.nsf/project.xsp?action=openDocument&name=JAVADDIN
- OSGi Tasklets (a.k.a DOTS)
  Track 4, Session 3: Let me introduce you to DOTS, Frank van der Linden
- ???

Quellen

Quellen

Bob Balaban:

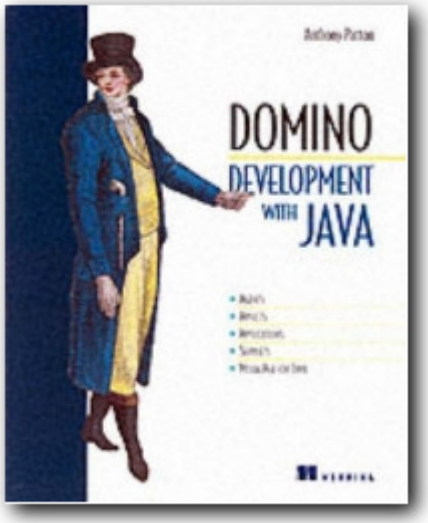
„Programming Domino 4.6 With Java“
Hungry Minds Inc, ISBN 1558515836, 465 S.



Web-Seite: http://bobzblog.com/tuxedoguy.nsf/dx/programming-domino-with-java-final-flush

Quellen (forts.)

Anthony S. Patton:
„Domino Development with Java“
Manning, ISBN 1-930110-04-9, 448 S.
(Jahrgang 2000, also noch Notes/Domino R5)



Web-Seite: http://www.manning.com/patton2/

Quellen (forts.)

„Programming Guide, Volume 3: Java/CORBA Classes“
IBM/Lotus, G210-2372-00, 1239 S.
(Jahrgang 2005, zum Domino Designer 7)

Web-Seite:
https://www.ibm.com/developerworks/lotus/documentation/dominodesigner/70x.html

Quellen (forts.)

Bruce Eckel:
„Thinking in Java“
Prentice Hall, ISBN 0-13-187248-6, 1.079 S.

Web-Seite: http://www.mindview.net/Books/TIJ/
(3. Auflage kostenlos als E-Book dort verfügbar)

Quellen (forts.)

Guido Krüger, Thomas Stark:
„Handbuch der Java-Programmierung“
Addison Wesley, ISBN 978-3-8273-2874-8, 1.356 S.

Web-Seite: http://www.javabuch.de/
(kostenlos als E-Book dort verfügbar)

Quellen (forts.)

Christian Ullenboom:

„Java ist auch eine Insel – Das umfassende Handbuch“
Galileo Computing, ISBN 978-3-8362-1506-0, 1.482 S.

Web-Seite: openbook.galileocomputing.de/javainsel9/
(kostenlos als E-Book dort verfügbar)

Quellen (forts.)

Java SE Tutorials:
http://download.oracle.com/javase/tutorial/

Java SE 6 Dokumentation:
http://download.oracle.com/javase/6/docs/

Java SE 6 API Specification:
http://download.oracle.com/javase/6/docs/api/

Alle Dokumentationen auch zum Download verfügbar

Quellen (forts.)

Want to debug Java agents INSIDE Designer? You can! (sort of):
http://bobzblog.com/tuxedoguy.nsf/dx/want-to-debug-java-agents-inside-designer-you-can-sort-of
http://www.ibm.com/developerworks/lotus/library/ls-Debugging_Java_agents/

OSGi Tasklets (a.k.a DOTS):
http://www.openntf.org/blogs/openntf.nsf/d6plinks/NHEF-8RXCN2

Fragen?

jetzt stellen – oder später:

tbahn@assono.de
http://www.assono.de/blog
04307/900-401

assono
IT-Consulting & Solutions

Folien unter:
www.assono.de/blog/d6plinks/EC13-Reite-den-Mustang

Und immer schön ans **recyclen** denken...

Zusatzmaterial

Agenda

- Einführung in die Programmiersprache Java
- Java in Lotus Notes & Domino
  - Applets
  - Agenten
  - Servlets
  - (Standalone-)Anwendungen
  - XPages
  - Plug-Ins
  - der Rest...

XPages

- eigentlich ist JavaScript „die“ Sprache der XPages
- Server-Side JavaScript (SSJS) wird im Hintergrund in Java übersetzt
- Java lässt sich einfach aus SSJS benutzen

XPages (forts.)

Demo: Hello World-XPage

Hello World-XPage

```
<?xml version="1.0" encoding="UTF-8"?>
<xp:view xmlns:xp="http://www.ibm.com/xsp/core">
   <xp:label value="Hallo EntwicklerCamp!" id="label1" style="font-
family:sans-serif;font-size:18pt;font-weight:bold"></xp:label>
   <xp:br></xp:br>
   <xp:text escape="false" id="computedField1">
      <xp:this.value><![CDATA[#{javascript:
var result : String = "Aktueller Benutzer:" +
                              session.getCommonUserName() + "<br />";
result += "Aktuelle Datenbank: " + database.getTitle() + "<br />";

var now : java.util.Date = new java.util.Date();
var formatter : java.text.SimpleDateFormat =
             new java.text.SimpleDateFormat(
             "'Heute ist 'EEEE', der 'dd.MM.yyyy'. " +
             "Es ist jetzt 'HH:mm' Uhr.'", Locale.GERMANY);
result += "Jetzt ist es: " + formatter.format(now) + "<br />";
return result}]]></xp:this.value>
   </xp:text>
</xp:view>
```

Demo:
Get Page By URL-XPage

Get Page By URL-XPage (Ausschnitt)

```
<xp:text escape="false" id="OpenedWebPage">
   <xp:this.rendered>...</xp:this.rendered>
   <xp:this.value><![CDATA[#{javascript:
importPackage(org.apache.http.client);
importPackage(org.apache.http.client.methods);
importPackage(org.apache.http.impl.client);
var urlToOpen : String = (sessionScope.get("URLToOpen") == null ?
                                   "" : sessionScope.get("URLToOpen"));
var result : String = "";
var client : HttpClient = new DefaultHttpClient();
try {
   var httpget : HttpGet = new HttpGet(urlToOpen);
   var handler : ResponseHandler = new BasicResponseHandler();
   var body : String = client.execute(httpget, handler);
   result = body;
} catch (e) {
   return "'" + urlToOpen + "' konnte nicht geöffnet werden."
} finally {
   httpclient.getConnectionManager().shutdown();
};
return result;}]]></xp:this.value>
</xp:text>
```

Get Page By URL-XPage (forts.)

- verwendet HttpComponents Client von Apache (unter Apache License v2): http://hc.apache.org/index.html
- jar-Dateien müssen bei XPages-Anwendungen in WebContent\WEB-INF\lib abgelegt werden, damit sie aus SSJS nutzbar werden (sichtbar nur in der **Java-Perspektive** des DDE)
- DDE & Notes-Client bzw. HTTP-Task **durchstarten** nach jeder Änderung!

Get Page By URL-XPage (forts.)

- Berechtigungen müssen ggf. in der Datei {Programmordner}\jvm\lib\security\java.policy vergeben bzw. erhöht werden; z. B.

```
grant {
    permission java.security.AllPermission;
};
```

- Beispiel ist **extrem unsicher**: Jeder darf alles!
- Details unter: http://download.oracle.com/javase/6/docs/technotes/guides/security/permissions.html
- DDE & Notes-Client bzw. HTTP-Task **durchstarten** nach jeder Änderung!

Agenda

- Einführung in die Programmiersprache Java
- Java in Lotus Notes & Domino
  - Applets
  - Agenten
  - Servlets
  - (Standalone-)Anwendungen
  - XPages
  - Plug-Ins
  - der Rest...

Plug-Ins

- seit Version 8 basiert Standard-Client auf Eclipse Rich Client Platform (RCP) und Lotus Expeditor
- im Wesentlichen eine "Sammlung" von Plug-Ins
- Erweiterungen über eigene Plug-Ins leicht möglich
- vorgebene Übergabepunkte: Extension Points
- eigene Plug-Ins können auch Extension Points anbieten
- Beispiele: Plug-Ins in der Seitenleiste, Toolbar-Buttons, Import-Filter, Datenquellen und Komponenten (Xpages) u.v.a.m.

IDE (forts.)

Demo: Eclipse mit Lotus Expeditor

wenn am Schluss noch Zeit bleibt...

Eclipse mit Lotus Expeditor

- für Plug-In-Entwicklung
  - Eclipse (3.4.2) installieren
  - Lotus Expeditor (6.2.2) installieren
  - Run Configuration für Notes erstellen
  - Plug-In schreiben
- beschrieben von Mario Gutsche in seinem Blog: http://www.mario-gutsche.de/2010/04/

Plug-Ins (forts.)

Demo: Hello World-Plug-In

Hello World-Plug-In (Ausschnitt)

```
public class MainViewPart extends ViewPart {

	private Label label = null;

	public MainViewPart() {
		super();
	}

	@Override
	public void createPartControl(Composite arg0) {
		this.label = new Label(arg0, SWT.NONE);
		label.setText("Hallo EntwicklerCamp!");
	}

	@Override
	public void setFocus() {
		label.setFocus();
	}

}
```

Hello World-Plug-In (Ausschnitt, forts.)

MANIFEST.MF

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Hello World  Plug-in
Bundle-SymbolicName: HelloWorldPlugIn;singleton:=true
Bundle-Version: 1.0.0
Bundle-Activator: de.assono.helloworldplugin.Activator
Bundle-Vendor: Thomas Bahn <tbahn@assono.de>
Require-Bundle: org.eclipse.ui,
 org.eclipse.core.runtime,
 com.ibm.rcp.ui
Bundle-RequiredExecutionEnvironment: JavaSE-1.6
Bundle-ActivationPolicy: lazy
```

Hello World-Plug-In (Ausschnitt, forts.)

plugin.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.4"?>
<plugin>
   <extension point="org.eclipse.ui.views">
      <view
            class="de.assono.helloworldplugin.views.MainViewPart"
            id="de.assono.HelloWorldPlugIn.views.MainViewPart"
            name="Hello World-Plug-in"
            restorable="true">
      </view>
   </extension>
   <extension point="com.ibm.rcp.ui.shelfViews">
      <shelfView
            id="de.assono.HelloWorldPlugIn.views.shelfView"
            region="TOP"
            showTitle="true"
            view="de.assono.HelloWorldPlugIn.views.MainViewPart">
      </shelfView>
   </extension>
</plugin>
```

Demo: Mindoo DDE Perspective Switcher-Plug-In

Mindoo DDE Perspective Switcher-Plug-In

- Details und Download im Mindoo-Blog:
  Free tool to quickly change perspectives in Domino Designer on Eclipse (DDE)
  http://www.mindoo.com/web/blog.nsf/dx/09.07.2009160821KLEJLA.htm

- Danke an Karsten Lehmann