In diesem Kapitel:

Java Deployment oder Wie werden Java Applikationen ausgeliefert?

1.1. Das Problem

Nach dem Sie viele Java Applikationen entwickelt haben, sollten Sie wieder etwas Geld verdienen. Daher stellt sich die Frage, wie man am Besten die Software beim Kunden installiert und eventuell wartet!

Auf dem Markt sind viele Installer vorhanden. Aber was muss konkret alles installiert werden?

Welcher Installer ist besser als die anderen, besser als ...?

Muss man auf kommerzielle Installer ausweichen?

Wird ein exe erzeugt?

Was muss sonst noch so alles installiert sein?

1.2. Einfachste Lösung

Wie könnte es anders sein:

die einfachste Lösung ist die JAR (Java Archive) Datei.

Aber:
es gibt Finessen. Sie können ein sogenanntes executable JAR generieren. Dieses ist dann auch ausführbar, sofern

- Sie auf dem Rechner die Java Runtime installiert haben (oder sogar JDK)
- Sie in der Registry (für alle die es gerne riskant lieben) oder im Explorer (für alle anderen) den Dateityp jar passend definiert haben.

Schauen wir uns diese Lösung genauer an.

1.2.1. Das JAR Dateiformat

Das Java ™ Archiv (JAR) Format gestattet es Ihnen mehrere Dateien (typischerweise Class Dateien) in ein einziges Archiv zu verpacken. Neben den Class Dateien enthält ein Archiv in der Regel alle benötigten Ressourcen, um die Applikation oder das Applet starten zu können.

Ab JDK Version 1.1 stand eine erste Version des JAR zur Verfügung. Dieses Format wurde in Version 2 angepasst und erweitert. Wir gehen nur auf die neuste Version ein!

1.2.1.1. Vorteile des JAR Dateiformates

Das JAR Format wurde in Anlehnung an ZIP definiert. Die Vorteile des Formats ergeben sich somit aus jenen des ZIP Formats:

- *Sicherheit*: Sie können ein Archiv digital signieren. Damit können Sie Zugriffsrechte auf den Inhalt des Archives definieren.
- *Reduzierte Ladezeit*: falls Sie beispielsweise ein Applet in ein JAR verpacken, können Sie das Applet schneller über das Web zum Client herunter laden. Zudem benötigen Sie nur eine einzige HTTP Verbindung: die Klassen müssen nicht einzeln herunter geladen werden.
- *Verdichtung*: das JAR Format gestattet Ihnen eine effizientere Speicherung der (zusammengehörenden) Klassen.
- *Packaging for Extensions* (Version 1.2): das Extension Framework aus der JAR Version 2 gestattet es Ihnen zusätzliche Funktionalitäten in ein JAR Archiv zu stecken. Das Feature wird zur Zeit jedoch kaum eingesetzt.
- *Package Sealing* (Version 1.2): Sie haben mit der neusten JAR Version die Möglichkeit Ihre Archive zu versiegeln. Damit erhöhen Sie die Sicherheit Ihrer ausgelieferten Software.
- *Package Versioning* (Version 1.2): neu kann im Archiv auch Versionsinformation abgespeichert werden. Damit haben Sie die Möglichkeit die Software Versionen zu kontrollieren.
- *Portabilität*: das JAR Dateiformat gilt für alle Java Plattformen.

1.2.2. Einsatz von JAR Dateien - Grundlagen

JAR Archive packen Dateien im ZIP Format. Damit kann man also all das, was man mit ZIP Dateien auch kann mit Java Archiven etwa gleich umsetzen:
packen, entpacken, auflisten, löschen, modifizieren, ...

Bevor Sie sich überlegen, wie Sie das Archiv elektronisch signieren wollen, sollten Sie verstehen, wie diese Grundoperationen ausgeführt werden können.

Java stellt Ihnen hierfür das Kommandozeile (schon wieder!) Werkzeug `JAR.EXE` zur Verfügung. Meines Wissens können Sie aber auch einfach mit WinZIP arbeiten und am Schluss noch das Archiv umbenennen.

Hier eine Kurzreferenz für das JAR Tool:

Operation	Befehl
Kreieren eines JAR Archives	`jar cf jar-file input-file(s)`
Anschauen des Inhalts eines Archives	`jar tf jar-file`
Extrahieren des Inhalts eines Archives	`jar xf jar-file`
Extrahieren einer bestimmten Datei aus dem Archiv	`jar xf jar-file archived-file(s)`
Ausführen eines als JAR Archives gespeicherten Java Programmes (Version 1.1)	`jre -cp app.jar MainClass`
Ausführen einer als JAR Archiv gespeicherten Java Applikation (Version 1.2 -- zusätzlich mit einem `Main-Class` Manifest Header)	`java -jar app.jar` `javaw -jar app.jar`
Starten eines Applets aus einem Archiv	`<applet code=AppletClassName.class archive="JarFileName.jar" width=width height=height> </applet>`

1.2.3. Starten ein Applikation in einem JAR Archiv

Wir betrachten mehrere Szenarien:

1) das JAR Archiv enthält ein Applet, welches in einem Browser gestartet werden soll.
2) das JAR Archiv enthält eine Applikation, welche auf der Kommandozeile gestartet werden soll.
3) Ihr Archiv enthät Programmcode, der eine andere Anwendung erweitert.

Den dritten Fall brauchen wir kaum zu besprechen: Sie können im Classpath, beim Starten der Applikation angeben, ob und welche JAR Archive eingebunden werden sollten.

1.2.3.1. Java Applets in JAR Dateien

Damit Sie ein Applet aus einer HTML Seite heraus starten können, müssen Sie dieses in einem Applet Tag angeben. Falls das Applet in einem Archiv zusammengefasst vorliegt, können Sie mit dem `Archive` Parameter angeben, wo das Archiv steckt, von wo es heruntergeladen werden soll.

Hier ein Beispiel:

- zuerst ohne Archiv:

```
<applet code=MeinWahnsinnsApplet.class
        width=120 height=120>
</applet>
```

- falls die Applet Dateien in einem Archiv zusammengepackt sind:

```
<applet code=MeinWahnsinnsApplet.class
        archive="Wahnsinn.jar"
        width=120 height=120>
</applet>
```

Der Archive Parameter spezifiziert den *relativen* Pfad zum JAR. Falls die HTML Datei und das Archiv im selben verzeichnis sind, brauchen Sie keine verzeichnisangaben zu machen.

Andernfalls könnte der Archive Parameter beispielsweise folgendermassen aussehen:

```
<applet code=VersteckDich.class
        archive="versteck/MeineArchive.jar"
        width=120 height=120>
</applet>
```

1.2.3.2. JAR Dateien als Applikationen

Ab Version 1.2 können Sie sogenannte ausführbare Archive definieren. Diese lassen sich mit dem Befehl:

```
java -jar jar-file
```

starten. Der Zusatz `-jar` zeigt dem Java Interpreter (der JVM), dass die Datei gepackt wurde.

Zusätzlich muss im Archiv mitgegeben werden, welche Klasse die Start Methode `main()` enthält. Dies geschieht mit einem

`Main-Class` Header

in der Manifest Datei (siehe unten):

Syntax:

```
Main-Class: classname
```

Der Wert `classname` ist der Name der Klasse, welche den Entry Point, die `main()` Methode enthält. Das Manifest können Sie mit dem `-m` Flag des Jar Utilities generieren lassen.

Dazu müssen Sie in einer Textdatei den Header eintragen und beim Archivieren als Parameter verwenden.

Beispiel:

- Sie haben ein Programm, `MeineApplikation.class` mit zusätzlich weiteren Klassen.
- Die Methode `main()` befindet sich in der Datei `MeineApplikation.class`
- Sie generieren eine Textdatei `meineApplikation.txt` , welche folgende Zeile enthält: `Main-Class: HelloWorld`
- nun können Sie archivieren:

```
jar cmf meineApplikation GISBroker.jar MeineApplikation.class
```

Nun können Sie Ihren GIS Broker als Java Applikation starten:

```
java -jar GISBroker.jar
```

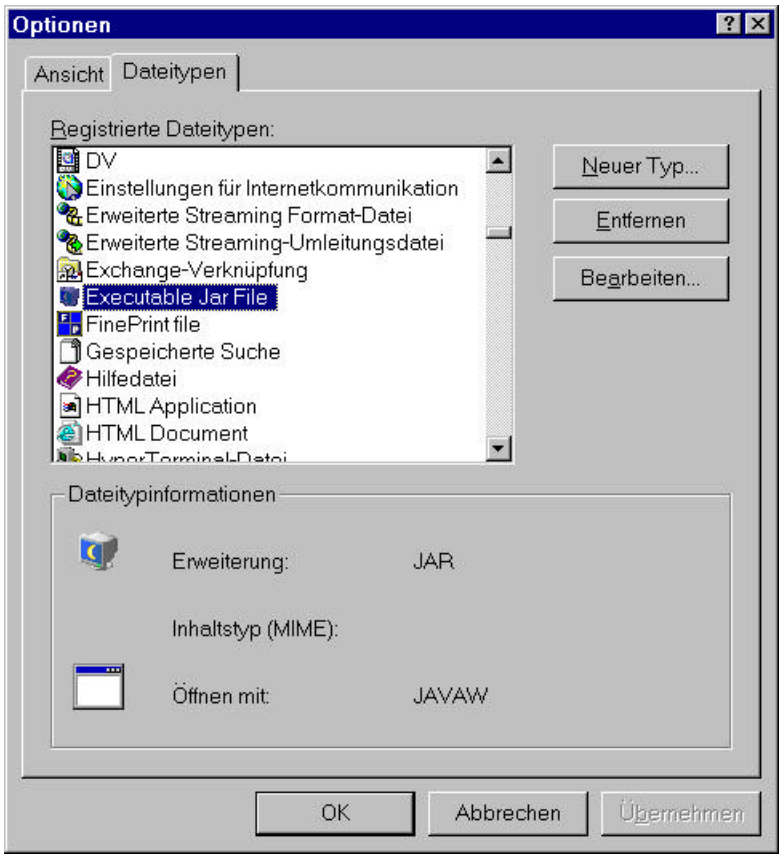
oder

```
javaw -jar GISBroker.jar
```

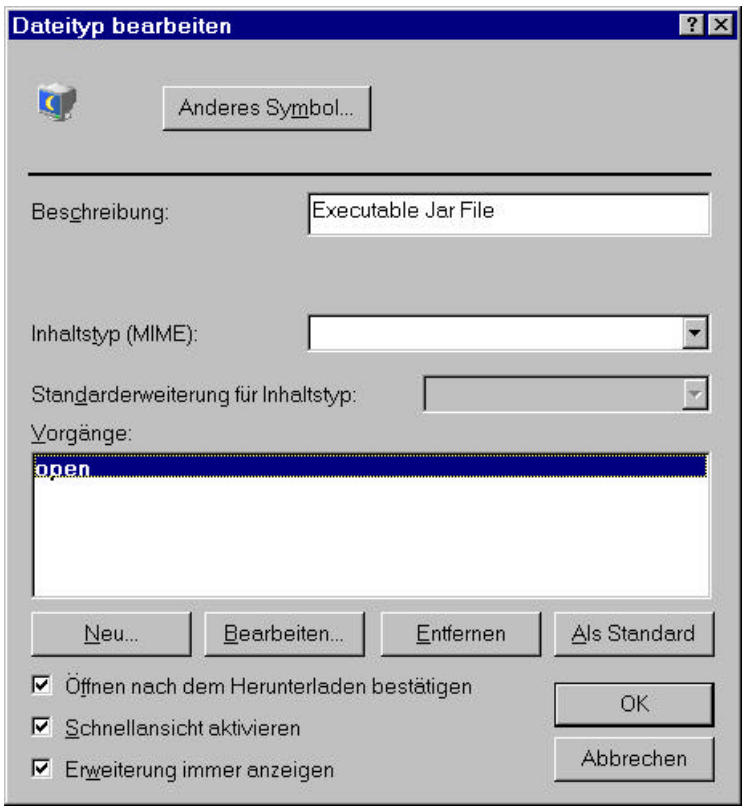
Das ist aber noch nicht so toll!
Sie möchten einfach nur auf das Archiv klicken, und dann soll die Applikation im Archiv, mit Manifest und allem drum und dran, starten!

Dazu müssen Sie einige Definitionen tätigen. Im Folgenden sehen Sie, wie diese in Windows NT aussehen. Die anderen Windows Versionen gestatten sicher ähnliche Verfahren.

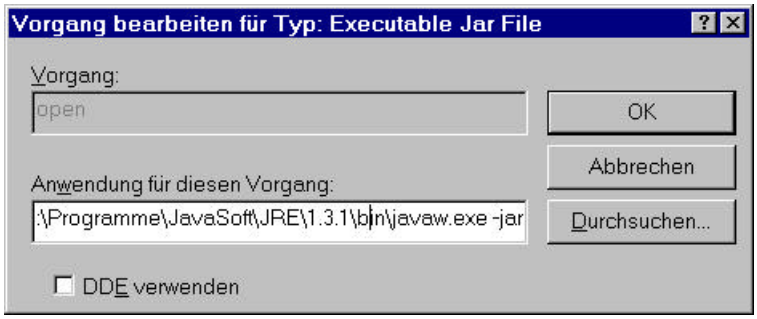
1) Definition des `Executable Jar File` Dateityps
Im Explorer (Dateiexplorer, nicht Internet Explorer) finden Sie unter Ansicht -> Optionen -> Dateitypen eine Liste der vordefinierten Dateitypen auf Ihrem Rechner.



Diese Angaben können Sie bearbeiten, oder neu erfassen.



Wesentlich ist eigentlich hier nur der open Befehl:



Dort wird angegeben, wie bzw. mit welchem Befehl das Archiv geöffnet werden soll. In unserem Fall ist dies

```
javaw.exe -jar
```

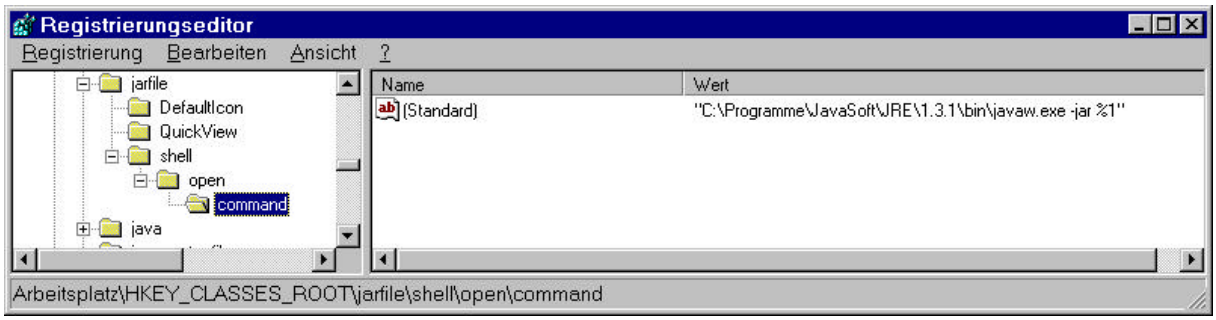
(dahinter folgt das Archiv).

Nun sind Sie dabei! Klicken Sie auf ein korrekt erstelltes executable Java Archiv wird die darin enthaltene Klasse mit der main-Methode gestartet!

Sie finden auf dem Server / der CD ein Beispiel dazu, in Form einer einfachen Swing Applikation. Konsolenapplikationen habe ich nicht näher untersucht!

2) Falls Sie auch mit den oben sichtbaren Angaben den Eintrag im Explorer nicht erstellen können, bleibt Ihnen der Weg über die Registry:

a) starten Sie den Registry Editor :
   unter Start : `regedit`
b) suchen Sie, ob `jar` schon irgendwie definiert ist
   Sonst machen Sie einen neuen Eintrag unter




```
HKEY_CLASSES_ROOT
jarfile
   Defaulticon
   QuickView
   shell
         open
               command
         (Standard) C:\Programme\JavaSoft\JRE1.3.1\bin\javaw.exe -jar %1
```

Natürlich muss der Pfad den Einstellungen auf Ihrem PC entsprechen!

Jetzt sind auch Sie dabei!

1.2.4. Vorgehen zum Installieren der Software beim Kunden

Damit haben wir ein Vorgehen, zur Installation der Java Software beim Kunden, welches fast ganz manuell wäre, also fehleranfällig und mühsam:

1) erstellen eines Java Archives mit einem passenden Manifest.
   Dazu können Sie JBuilder einsetzen!
2) beim Kunden:
   - Prüfen, ob auf der Maschine JDK oder JRE vorhanden ist
   - Definieren der Zuordnung des Dateityps jar zu javaw.exe im JRE
   - kopieren des Archives und Zuordnung eines Icons und eventuell einer Programmgruppe

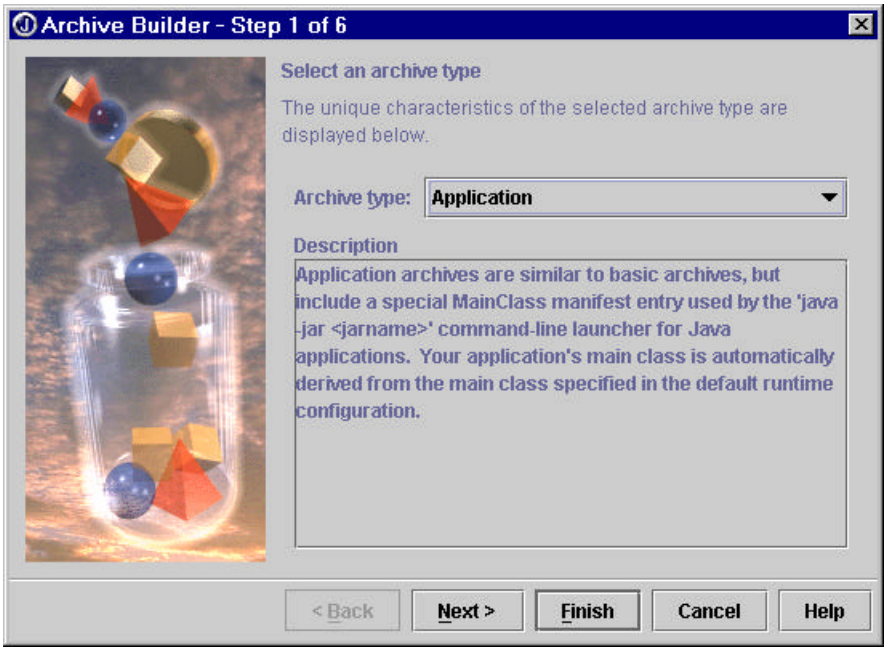
Klingt mühsam und ist es eigentlich auch. Aber für Inhouse Installationen ist diese Variante brauchbar, weil bei Ihnen sicher JRE *und* JDK installiert sind.

1.3. Erstellen von ausführbaren Archiven im JBuilder

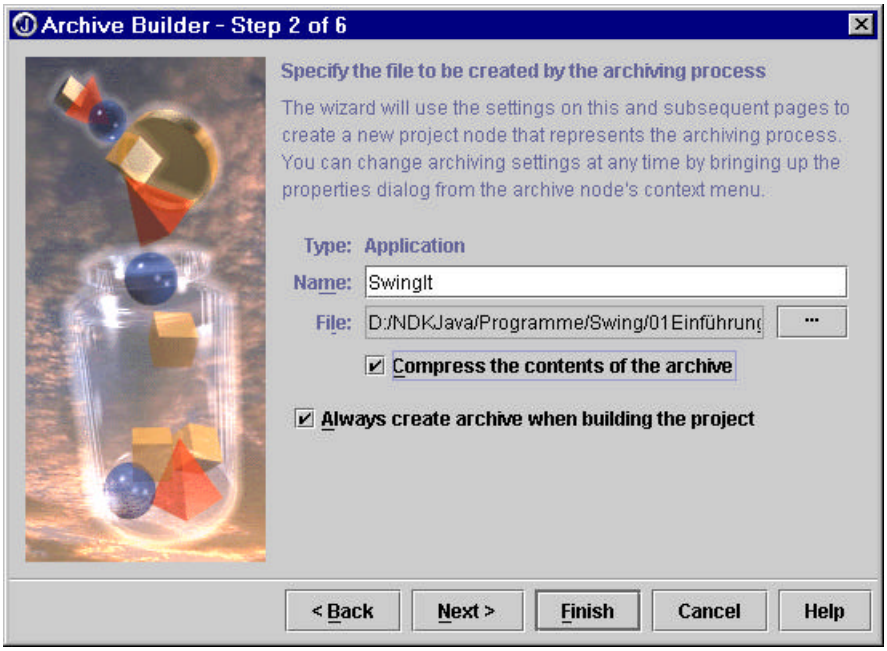
Der JBuilder und andere IDEs stellen Ihnen einige Wizzards zur Verfügung, mit deren Hilfe Sie executable Jar Dateien erstellen können, inklusive Zusammensuchen der benötigten Klassendateien.

Schauen wir uns ein Beispiel an.

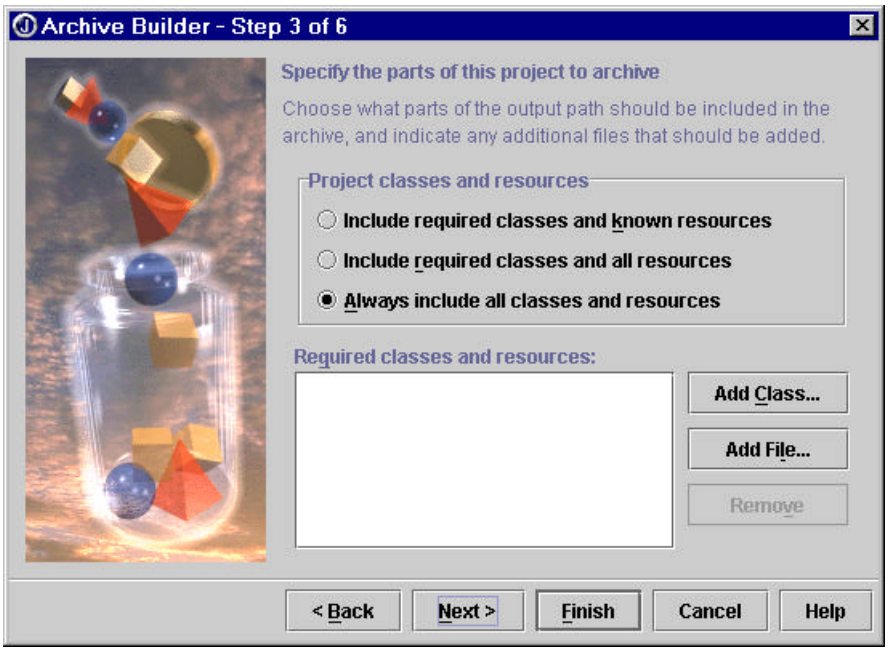
- Öffnen Sie den `JBuilder`, sinnvollerweise mit einem Projekt, welches eine grafiische Oberfläche enthält.
- Unter Wizards finden Sie den `Archive Builder`
  Der JBuilder meldet sich mit einem Fenster. Wählen Sie die Option `Application`:



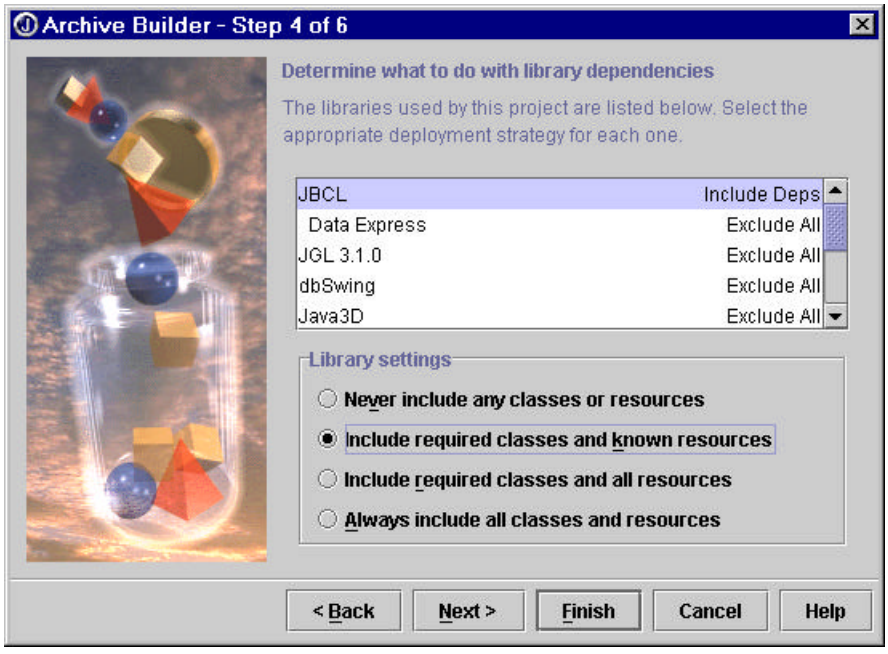
- im nächsten Fenster können Sie den Namen des zu generierenden Archives angeben:



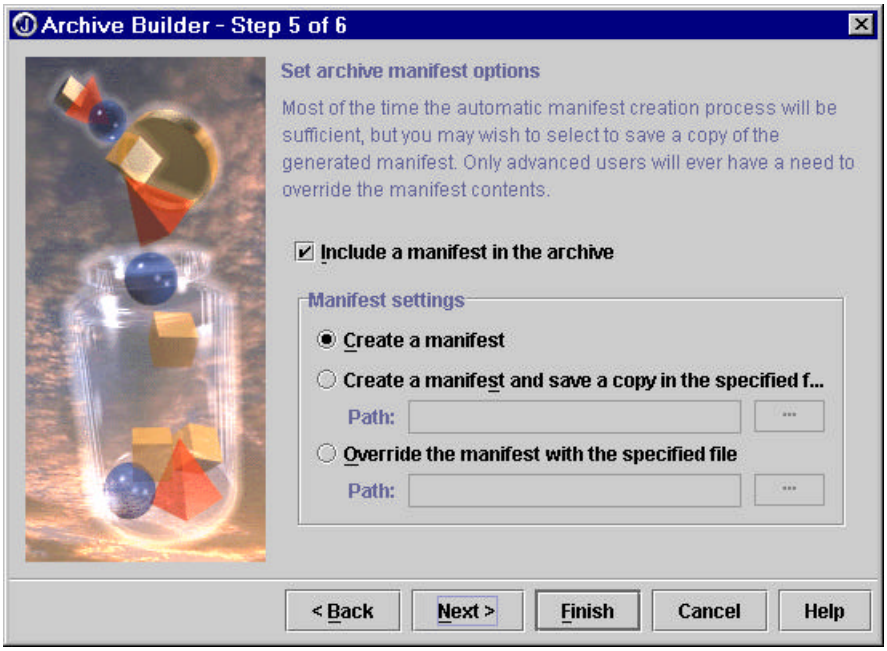
- dann binden wir alle Klassen und Ressourcen ein:



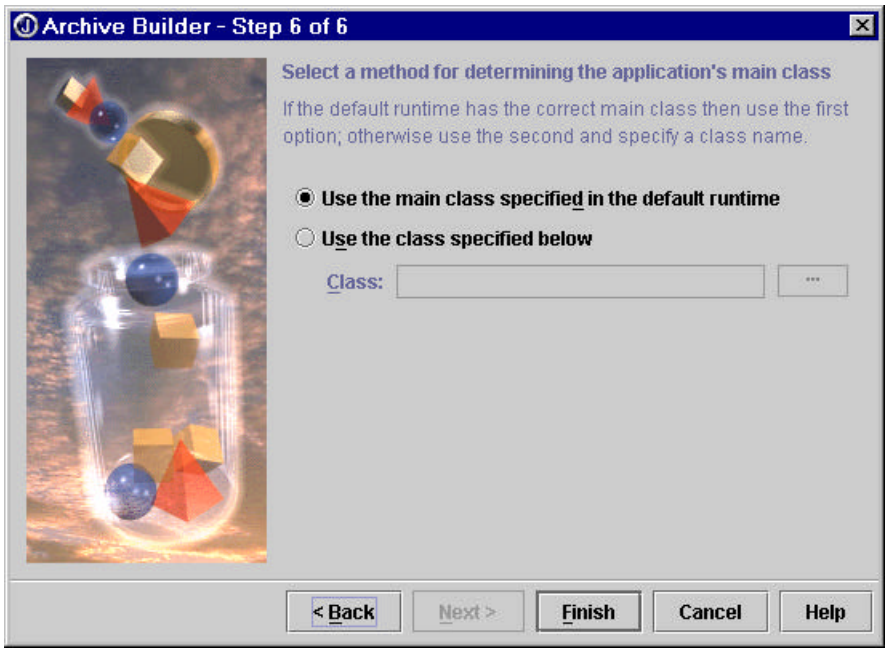
- und nur jene Klassen und Ressourcen aus den Bibliotheken, die wir echt benötigen:



- die Manifestdatei, damit das Archiv ausführbar werden kann:



- den Eintrag im Manifest soll JBuilder passend ergänzen, mit der Angabe aus dem Run im JBuilder (Angabe der Klasse mit der `main()` Methode).



Jetzt sind Sie fast fertig.

Der JBuilder generiert Ihnen einen Eintrag im Projektfenster. Mit `make` wird daraus das Archiv (oder beim nächsten Rebuild des Projektes).

Sie können nun auch das Archiv im JBuilder aufmachen und werden das Manifest sehen, in meinem Fall:

```
Manifest-Version: 1.0
Main-Class: SwingBeispiel.SwingApplikation
```

Damit haben wir unser Java Archiv im Projektverzeichnis.

Testen Sie das Archiv, indem Sie das in der Manifest Datei spezifizierte Programm durch Doppleklick starten.

1.4. Der Installer

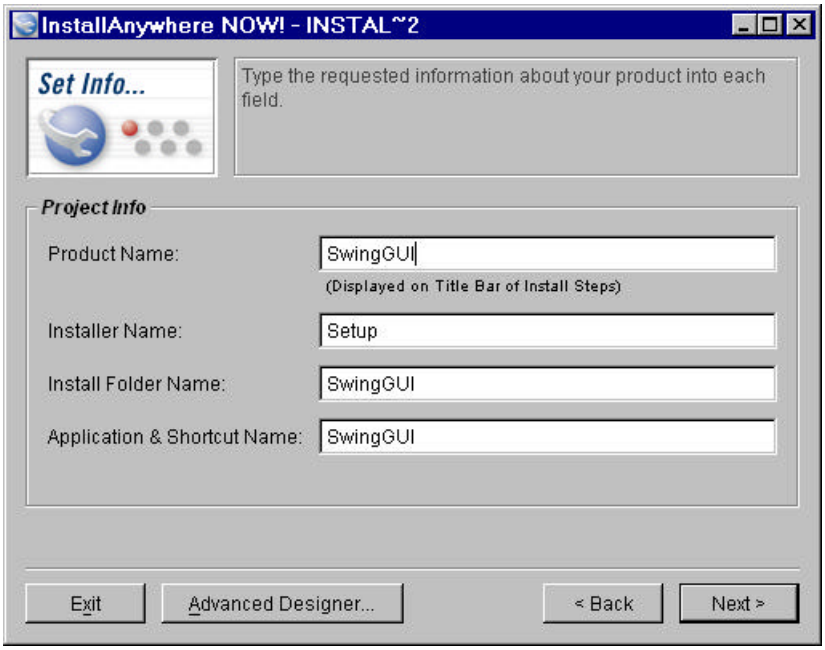
Nachdem alle wissen, dass es viele unterschiedlichen Installer gibt, ist eine umfassende Beschreibung des für Sie optimalen Installers.

Allerdings:
bei ZeroG können Sie einen in Java geschriebenen Installer, InstallAnyWhere, herunterladen. Dieser ist in der kleinsten version auch gratis. Diese version ist zum Testen nett, für den produktiven Einsatz empfiehlt sich die mächtigere Version, welche auch Install.exe mit allem drum und dran generiert.

Hier also eine kurze Übersicht über InstallAnyWhere 4 Now. Dieses Programm kann bei Ihnen auf dem PC installiert werden. Es führt Sie durch die Generierung eines Install.exe mit den von Ihnen angegebenen Dateien, Programmen und was auch immer.

Hier ein Beispiel:

Wir wollen unser ausführbares Archiv auf unserem PC installieren. Dazu wollen wir einen Installer verwenden.

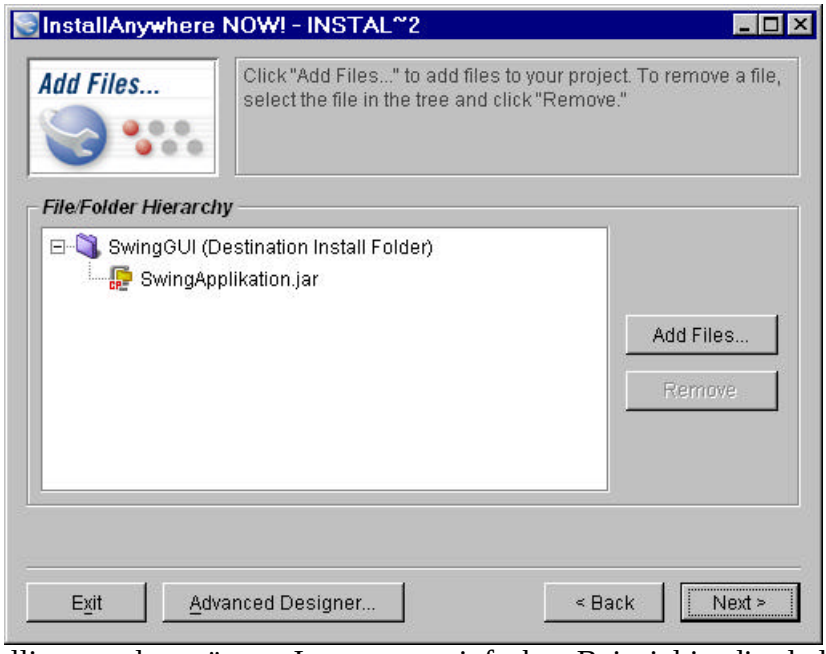


Der Installer führt Sie durch den Prozess.Allerdings haben Sie wesentlich mehr Möglichkeiten Ihr Setup Ihren Bedürfnissen anzupassen, beispielsweise eine VM zu suchen oder auszuwählen. Zuerst betrachten wir kurz den Ablauf beim Wizzard und gehen anschliessend auf das Problem JDK versus JRE versus weitere JVMs ein. Auf vielen Rechnern befindet sich bereits ein JDK oder ein JRE, also die Laufzeitumgebung für Java (JDK ist für die Entwickler: Java Development Kit). Der Speicherbedarf der zwei Systeme ist auch wesentlich verschieden:

JDK benötigt :
72 MB plus 109 MB Dokumentation (API, User Guides, ReadMe)

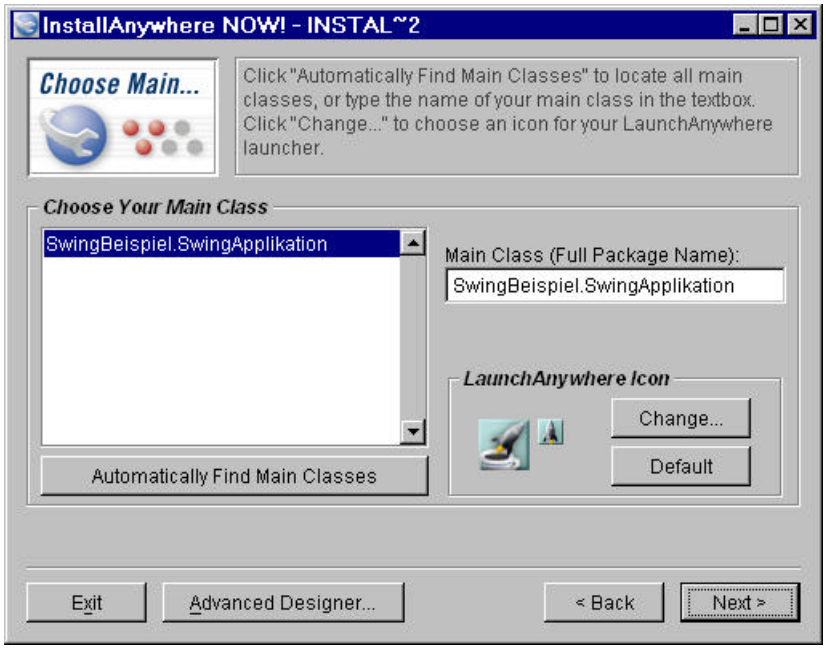
JRE dagegen kommt mit
27 MB aus

Der Installer benötigt eine Liste der Klassen oder Archive bzw. Bibliotheken, welche für die

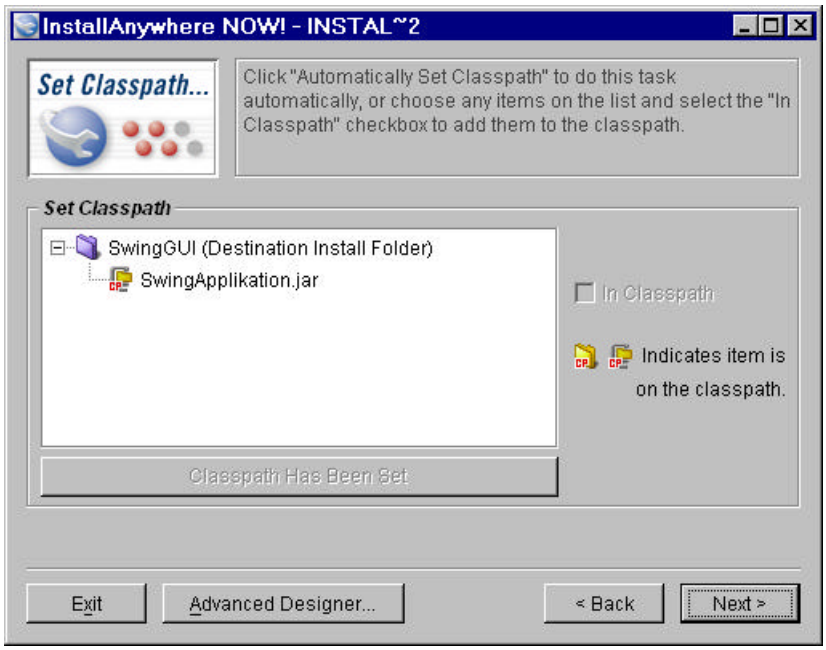


Applikation installiert werden müssen. In unserem einfachen Beispiel ist dies lediglich das ausführbare Archiv.

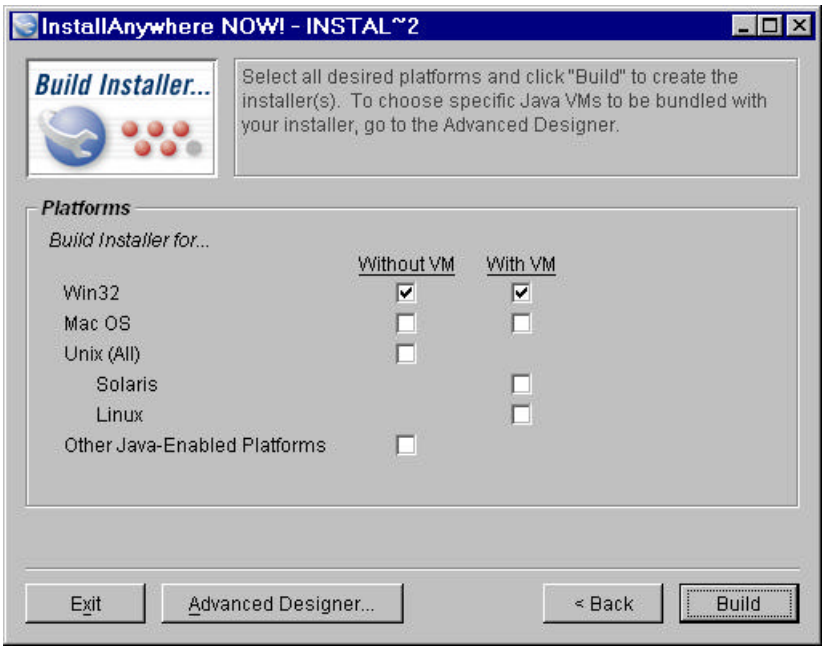
Da wir bei der Definition des ausführbaren Archives die Klasse mit der main() Methode angegeben haben, findet der Installer diese Information aus der Manifest-Datei:



Auch der Klassenpfad wird vom Wizzard gesetzt, da in unserem Fall keine speziellen Angaben benötigt werden:



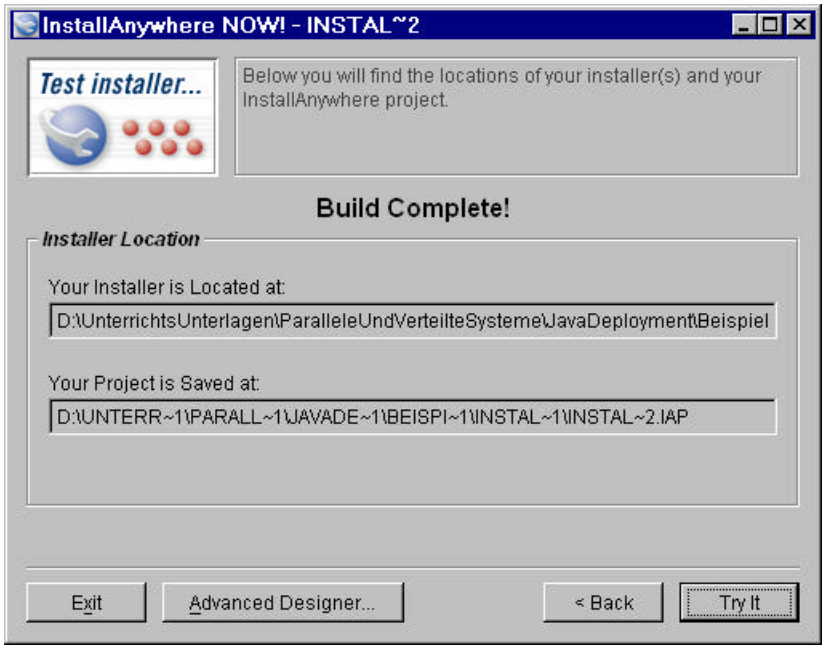
Jetzt können Sie angeben, ob ein Setup Programm für mehrere Plattformen mit oder ohne JVM kreiert werden soll.



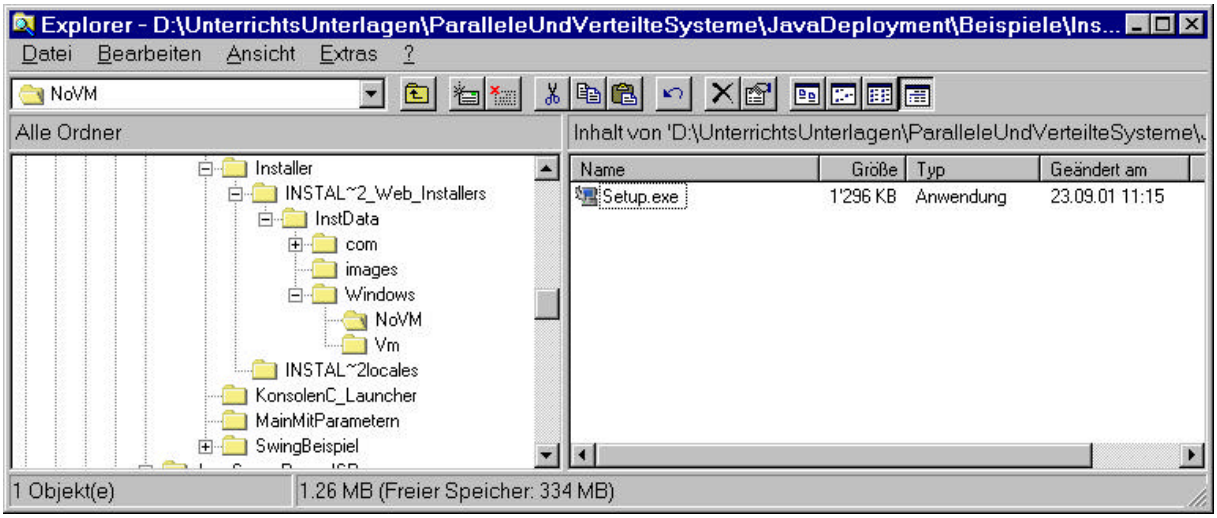
Hier haben wir bewusst zwei Optionen angewählt, zum einen um zeigen zu können, wieviel zusätzlicher Platz die VM benötigt; zum anderen, um zeigen zu können, wie diese unterschiedlichen Versionen abgelegt werden.

Damit ist die Arbeit mit dem Wizzard abgeschlossen!

Sie können das Setup Programm gleich testen:



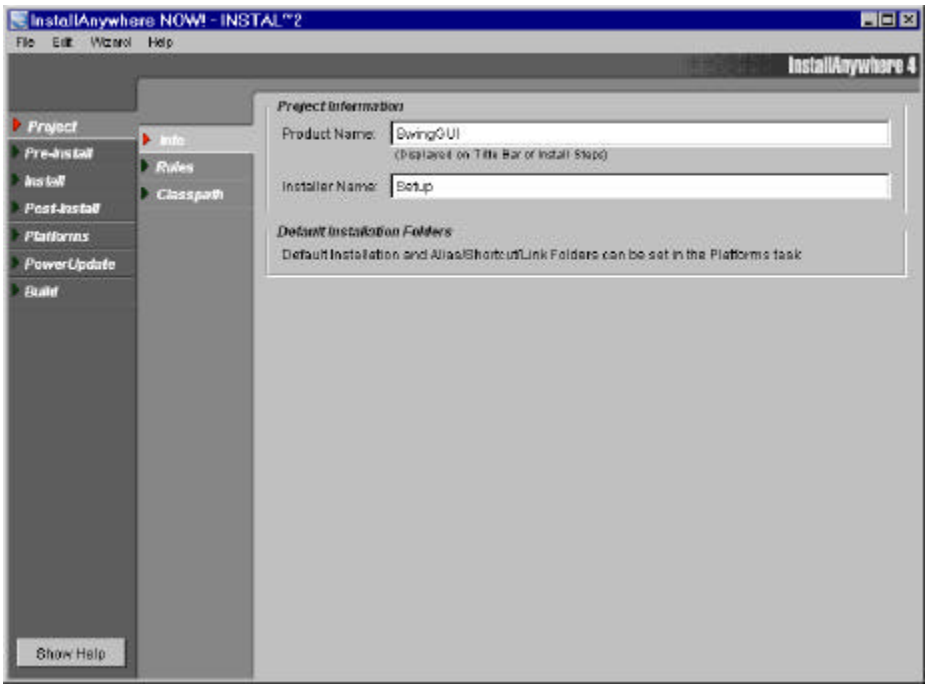
Sie finden in den Verzeichnissen ...InstDat\Windows\NoVM bzw. ...\Vm



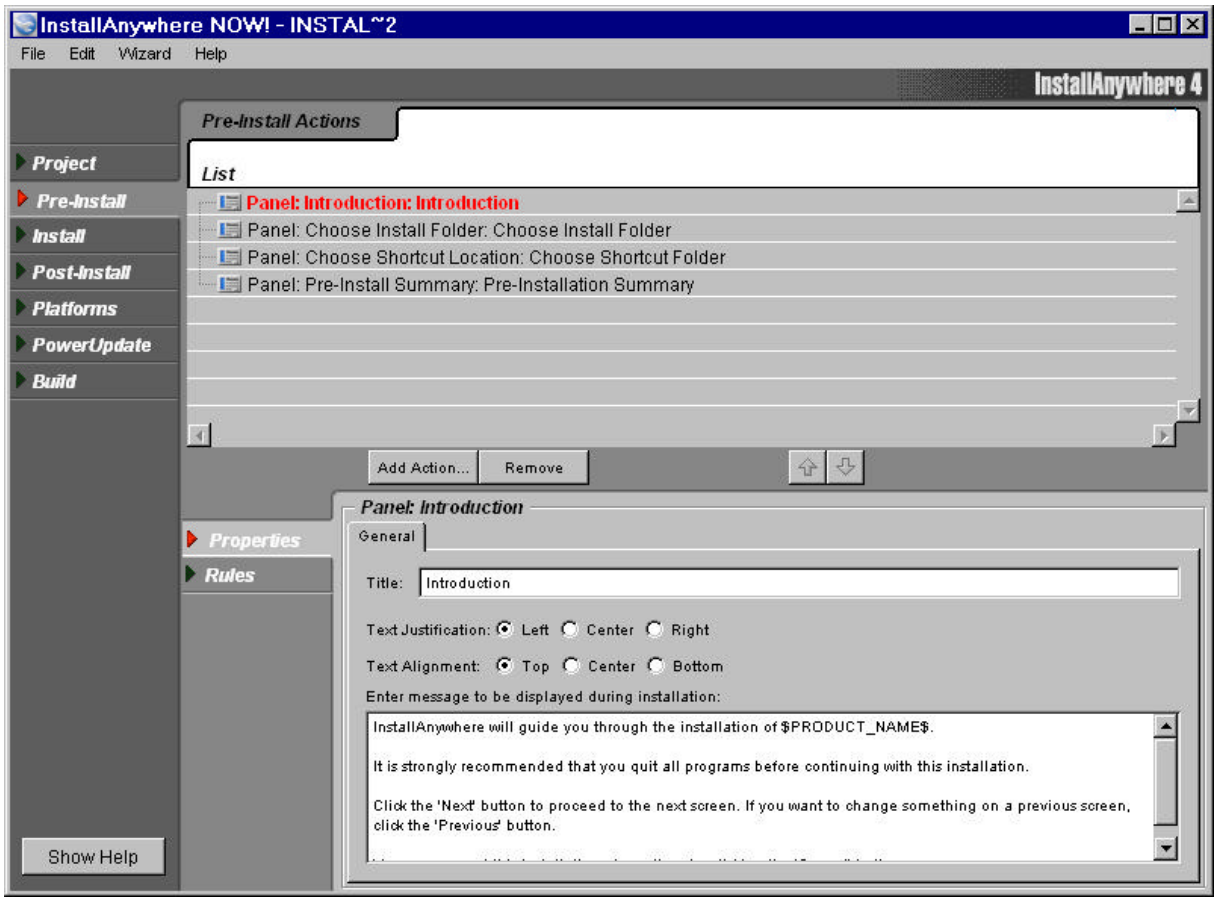
Der Speicherbedarf (inklusive Archiv bzw. Klassendateien) beträgt:

- ohne VM: ca 1.9 MB
- mit VM : ca 5.8 MB

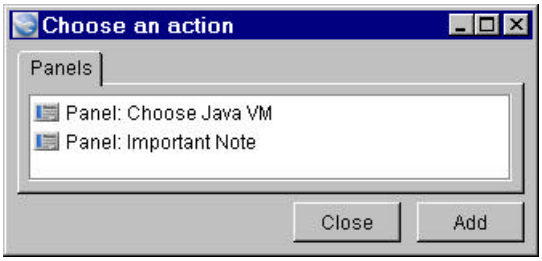
Wir wechseln gleich in den Designer, um gezielter Ergänzungen und Erweiterungen ins Setup Programm einbauen zu können:



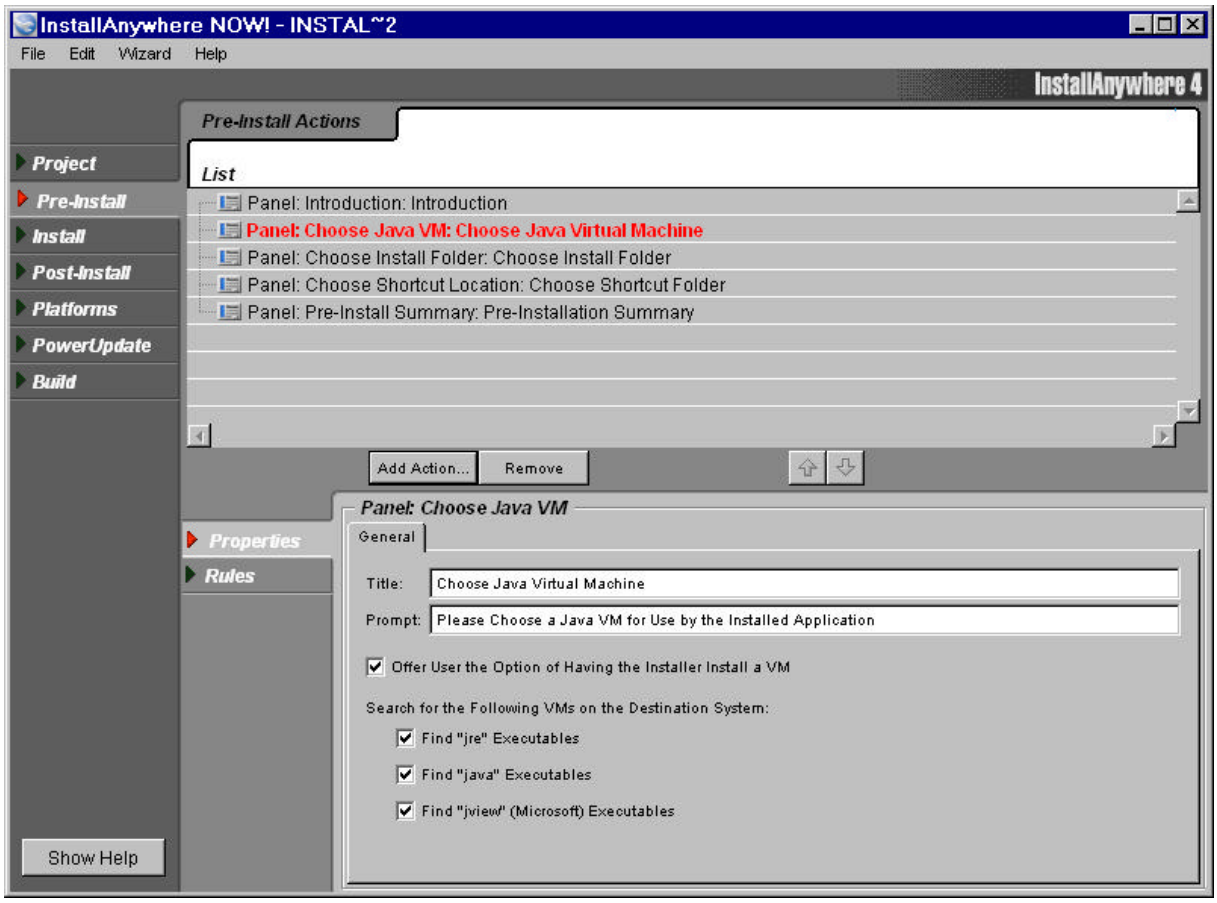
Wir wollen speziell vor der Installation die bereits vorhandenen VMs finden und eine aussuchen. Dafür müssen wir im Bereich 'Pre-Install' eine weitere 'Action' einfügen:



Wenn Sie auf die Auswahl 'Add Action...' (Mitte des Fensters) anklicken, öffnet sich ein weiteres Fenster:



Damit ändert sich die Auswahl im Hauptfenster:

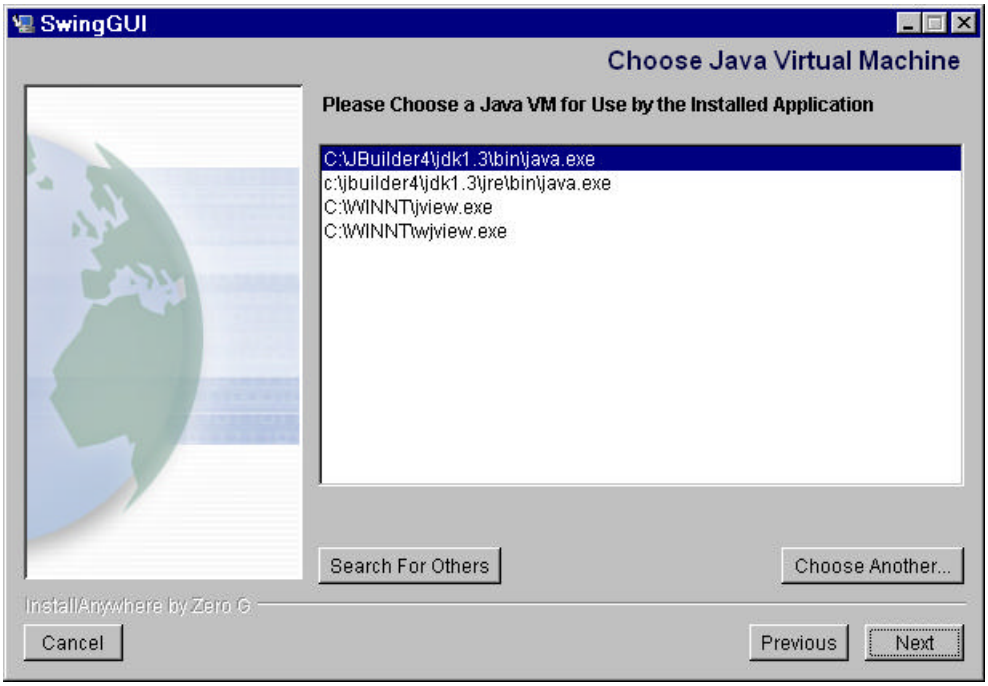


Jetzt können wir wieder zum Wizzard zurück wechseln (Wizzard anwählen, oben in der Menüliste) und einfach durchklicken oder diese Version unter einem anderen Namen abspeichern.

Falls Sie die Version nicht unter einem neuen Namen abspeichern, wird die alte Version überschrieben, speziell die Version mit der VM gelöscht.

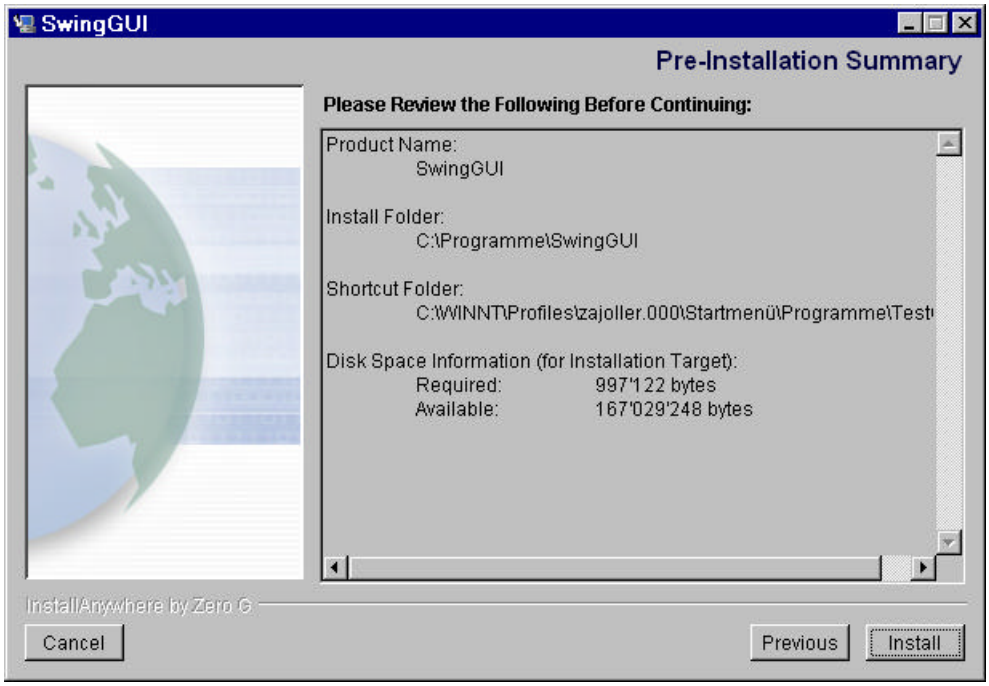
Sie können jetzt mit dem Setup Programm eine Installation bei Ihrem Kunden ausführen. Den Text, welcher vom Setup Programm angezeigt wird (und das Hintergrundbild) können Sie ebenfalls anpassen.

In unserem Fall sucht das Programm zuerst im Drive C: nach JVMs:



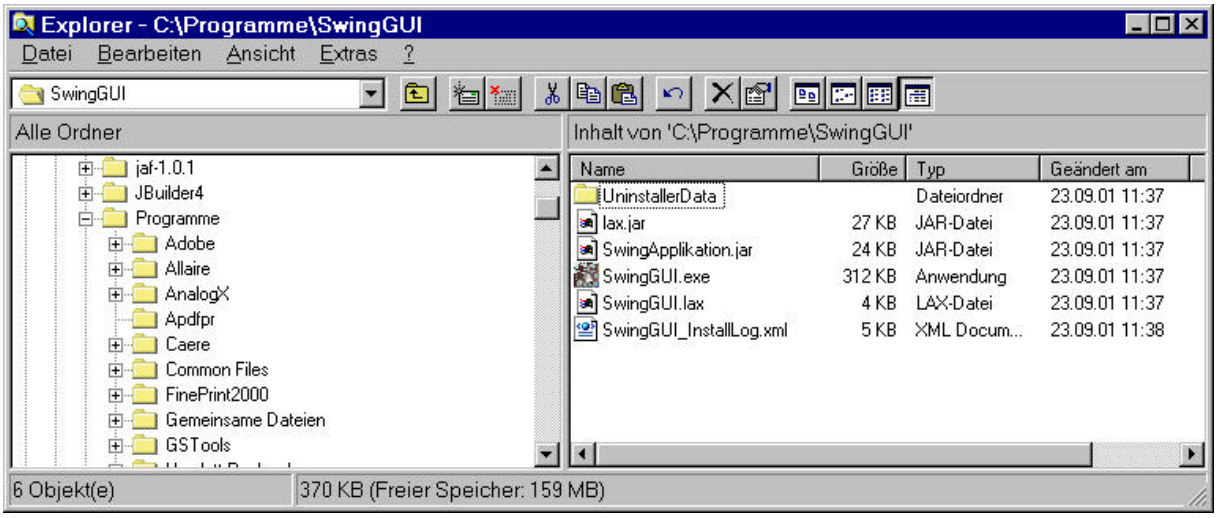
Falls Sie die Option 'Choose Another' anwählen, können Sie weitere von Hand weitere JVMs suchen oder einbinden.

Das Setup Programm zeigt Ihnen einige weitere Fenster, in denen Sie Optionen, wie etwa Programmgruppe, Verzeichnis, Icons, ... auswählen können.

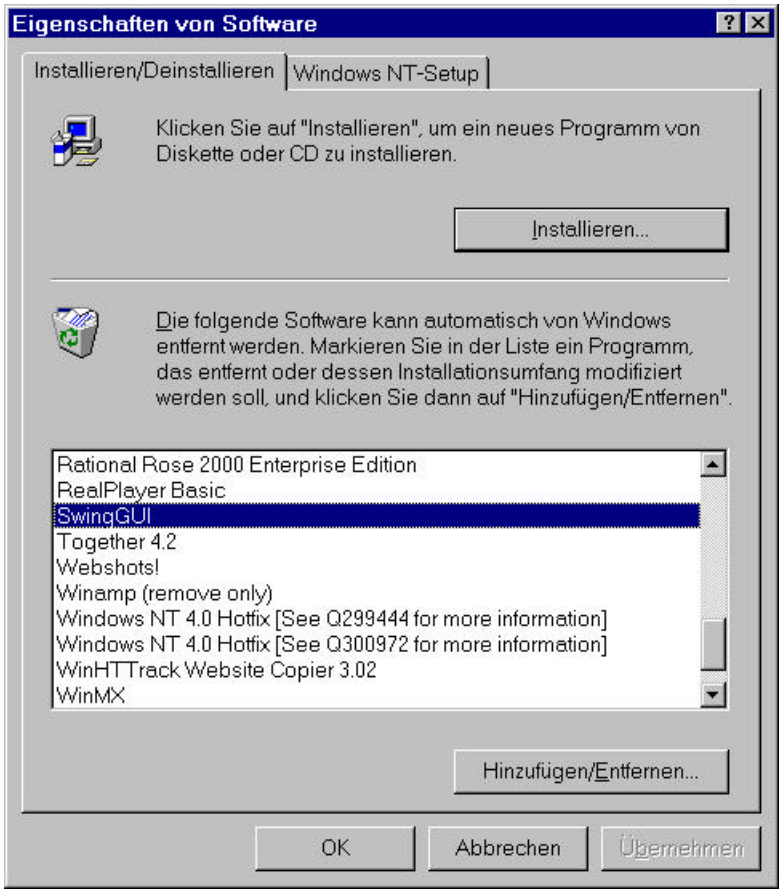


Nach diesem Fenster wird die Software installiert und die Programmgruppe, ... angelegt.

In unserem Fall finden Sie im Programmverzeichnis folgende Dateien und Einträge:



Die exe Datei ist der Starter, mit dem die Klassen (hier das Archiv) geladen werden. Natürlich wurde die Software auch in die Registry eingetragen. Sie können Sie jederzeit in der Systemsteuerung unter Software finden und beispielsweise wieder entfernen.



Zusammenfassend kann gesagt werden, dass dies sicher eine sehr befriedigende Lösung für das Deployment, das Ausliefern der Software zum Kunden ist.

1.5. Die Geek Lösung

Hinweis: Geek Lösungen sind ausgeflippte Lösungen, die zeigen, was möglich ist. Aber Sie sollten sich überlegen, ob dies für Sie eine akzeptable Lösung ist!

Betrachten Sie einmal das Verzeichnis des Jbuilders:

- es wird Ihnen auffallen, dass mehrere Programme (exe) paarweise vorhanden sind:
  ein ppppW.exe und ein pppp.exe.
  Beispiel: JBuilder.exe, JbuilderW.exe

- zudem finden Sie in der Regel dazu passende Konfigurationsdateien
  Beispiel: JBuilder.config

- und in diesen Konfigurationsdateien eventuell Hinweise auf weitere Konfigurationsdateien, beispielsweise im JBuilder.config :

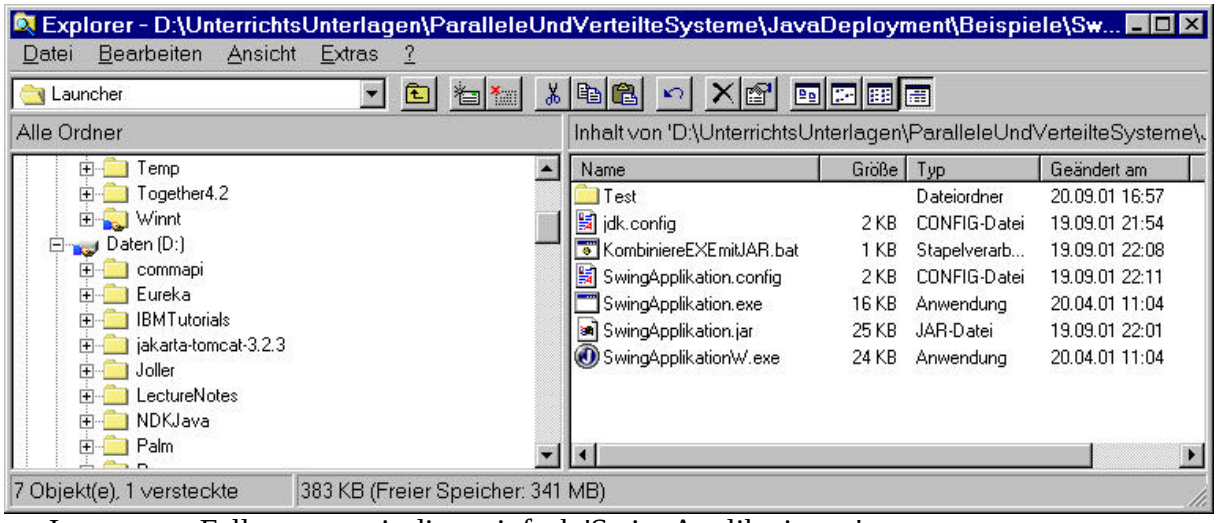
```
# Read the shared JDK definition
include jdk.config
```

Diese Paare sind Loader Programme mit einem sehr einfachen Aufbau, auf den wir noch (in C/C++) zurück kommen.

Der Trick besteht nun darin, diese Loader Programme für eigene Zwecke umzukonfigurieren und neu zu benennen:

- kopieren Sie die Dateien
  `JbuilderW.exe`
  `JBuilder.exe`
  `JBuilder.config`
  `jdk.config` und
  `launcher.dll`
  in ein separates Verzeichnis

- benennen Sie die JBuilder Dateien um



In unserem Fall nennen wir diese einfach 'SwingApplikation....'.

- Die Batch Datei dient dem Kreieren des eigenen Launchers:
  wir kopieren das ausführbare Archiv ins exe!

```
Rem kombinieren des EXE Launchers mit der Class Datei (dem Archiv)
Rem
COPY /B SwingApplikation.exe+SwingApplikation.jar SwingApplikation.exe
```

  Das Ganze klingt verrückt ('Geek Lösung') funktioniert aber! Hinter dem relevanten Teil des exe Programms ist noch genügend Platz!

- passen Sie die Pfade in den Konfigurationsdateien so an, dass die JRE bzw. JDK Bibliotheken gefunden werden!

- nun haben Sie einen eigenen Launcher.Vielleicht funktioniert der Trick in neuen Releases nicht mehr, also Vorsicht!

Das vollständige Beispiel finden Sie auf dem Server / der CD. Beachten Sie, dass ich aus Tippgründen eine Version für ein 'D:\Test' Verzeichnis erstellt habe. Sie sollten die Pfadangaben also auf jeden Fall anpassen:

In jdk.config:

```
# Add the "tools" JAR to the path since the JDK doesn't
# do so automatically
addpath C:/JBuilder4/jdk1.3/lib/tools.jar

# Use the embedded JDK provided with JBuilder Foundation
# and override the default heap growth / shrinkage rates
javapath C:/JBuilder4/jdk1.3/bin/java
vmparam -Xminf0.2
vmparam -Xmaxf0.2
```

in der Applikations-Konfigurationsdatei:

```
# Read the shared JDK definition
include jdk.config

# Tune this VM to provide enough headroom to work on large
# applications
vmparam -Xms8m
vmparam -Xmx128m

# Put the Light AWT wrapper on the boot path
addbootpath C:/JBuilder4/lib/lawt.jar

# WICHTIGE Zusatzzeile
addpath D:/Test/SwingApplikation.exe


# Add all JAR files located in the lib and lib/ext directory
addjars C:/JBuilder4/lib
addjars C:/JBuilder4/lib/ext
..

# Add all the configuration files located in the lib/ext directory
includedir C:/JBuilder4/lib/ext

# Start JBuilder using the its main class
mainclass SwingBeispiel.SwingApplikation
```

1.6. C / C++ Lösungen

Falls Ihnen die bereits vorgestellten Methoden für das Deployment noch nicht genügen oder Sie besser verstehen möchten, wie ein Launcher aufgebaut ist, können Sie im Folgenden mögliche Launcher in C kennen lernen.

Alle Beispiele verwenden das Java Native Interface (JNI). Sun liefert Ihnen mit dem JDK mehrere '.h' Dateien mit, welche alle relevanten Definitionen enthalten, die Sie für das Generieren des Launchers benötigen.

1.6.1. Konsolen Launcher

Im einfachsten Fall brauchen Sie einen Launcher für ein Konsolen Programm. So blöd das Programm auch ist, es zeigt, wie einfach aus C die VM gestartet werden kann:

```
#include <windows.h>
#include <stdio.h>
#define MAIN_CLASS  "HalloDuDa"
int main( int argc, char** argv )
{
  char* cmdline = malloc( 1024 ); /* big buffer */
  // kopieren des 'java..' Textes in den Buffer cmdline
  sprintf( cmdline, "java.exe %s", MAIN_CLASS );
  // ausführen der Kommandozeile
  system( cmdline );
  return 0;
}
```

Natürlich könnten Sie auch gleich die Kommandozeile eintippen!

1.6.2. VM Launcher

Im einfachsten Fall übergeben Sie keinerlei Parameter und möchten aber die JVM laden:

```
#include <windows.h>
#include <stdio.h>
#include <jni.h>
#define MAIN_CLASS  "HalloDuDa"
int main( int argc, char** argv )
{
  JNIEnv*        env;
  JavaVM*        jvm;
  JDK1_1InitArgs vmargs;
  jint           rc;
  jclass         cls;
  jmethodID      mainId;
  /* CLASSPATH environment variable bestimmen */
  char* szClasspath = getenv( "CLASSPATH" );
  vmargs.version = 0x00010001;  /* version 1.1 */
  JNI_GetDefaultJavaVMInitArgs( &vmargs );  /* init vmargs */
  /* classpath von JNI_GetDefaultJavaVMInitArgs ist falsch */
  vmargs.classpath = szClasspath;
  rc = JNI_CreateJavaVM( &jvm, &env, &vmargs );  /*  JVM kreieren*/
  if( rc < 0 )
  {
```

```
    printf("Kreieren der JVM schlug fehl\n");
    return 1;
  }
  /* laden der Klasse mit static main() Methode */
  cls = (*env)->FindClass( env, MAIN_CLASS );
  if( cls == 0 )
  {
    printf( "Die folgende Klasse konnte nicht gefunden werden %s\n",
MAIN_CLASS );
    return 1;
  }
  /*  main() Methode suchen */
  mainId = (*env)->GetStaticMethodID(env, cls, "main",
"([Ljava/lang/String;)V");
  if( mainId == 0 )
  {
    printf( "Die Methode main() wurde nicht gefunden\n" );
    return 1; /* error */
  }
  (*env)->CallStaticVoidMethod(env, cls, mainId, 0); /* call main() */
  (*jvm)->DestroyJavaVM( jvm );  /* kill JVM */
  return 0;
}
```

In diesem Programm verwenden wir einige der im JNI definierten Funktionen (in C), wie beispielsweise '`CallStaticVoidMethod(...)`'.

```
#include <windows.h>
#include <stdio.h>
#include <jni.h>
#define MAIN_CLASS  "HalloGUI"
int APIENTRY WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR
lpCmdLine, int nCmdShow)// GUI
{
  JNIEnv*        env;
  JavaVM*        jvm;
  JDK1_1InitArgs vmargs;
  jint           rc;
  jclass         cls;
  jmethodID      mainId;
  /* CLASSPATH environment variable setting */
  char* szClasspath = getenv( "CLASSPATH" );
  vmargs.version = 0x00010001;  /* version 1.1 */
  JNI_GetDefaultJavaVMInitArgs( &vmargs ); /* init vmargs */
  /* classpath von JNI_GetDefaultJavaVMInitArgs ist falsch */
  vmargs.classpath = szClasspath;
  rc = JNI_CreateJavaVM( &jvm, &env, &vmargs );  /* kreiere JVM */
  if( rc &lt; 0 )
    return 1;
  /* laden der Klasse mit static main() Methode */
  cls = (*env)->FindClass( env, MAIN_CLASS );
  if( cls == 0 )  return 1;
  /* main() Methode */
  mainId = (*env)->GetStaticMethodID(env, cls, "main",
"([Ljava/lang/String;)V");
  if( mainId == 0 )
    return 1; /* error */
  (*env)->CallStaticVoidMethod(env, cls, mainId, 0); /* call main() */
  (*jvm)->DestroyJavaVM( jvm ); /* kille die JVM */
  return 0;
}
```

Was ist schlecht an diesen Skizzen: es sind Skizzen.
Besser würden wir die Parameter, den Namen der Applikation ,... aus einer Datei (Config....) lesen. Aber die Skizzen zeigen, wie so etwas aussehen könnte, mehr nicht! Ich werde mich bemühen eine bessere Lösung auf dem Web verfügbar zu machen!

1.7. Web Start und das Java™ Network Launching Protocol & API?

1.7.1. Einführung

Java Web Start ist eine neue Technologie für den Einsatz im Zusammenhang mit Java™ Technologie-basierten Anwendungen. Damit wird die Verbindung zwischen dem Computer und dem Internet hergestellt, über die die Anwender Anwendungen direkt über das Internet starten und verwalten können. Java Web Start ermöglicht die einfache Aktivierung von Anwendungen mit einem einzigen Mausklick und garantiert, dass Sie immer die neueste Version ausführen. Dadurch werden komplizierte Installations- oder Aktualisierungsprozeduren vermieden.

Traditionell erfordert der Vertrieb von Software über das Internet, dass die Anwender das Installationsprogramm im Web finden und herunterladen, das Installationsprogramm auf dem Computer finden und ausführen müssen. Das Installationsprogramm fragt dann nach Installationsverzeichnissen und -optionen, z. B. volle, typische oder minimale Installation. Das ist in der Regel eine zeitaufwendige und komplizierte Aufgabe und muss für jede neue Software-Version wiederholt werden.

Im Gegensatz dazu sind *über das Internet bereitgestellte Anwendungen* wie Ihr bevorzugter HTML-basierter E-Mail-Client und Kalender, Auktionswebsites usw. leicht zu installieren und anzuwenden. Durch den Web-Browser läuft der gesamte Prozess automatisch ab. Es gibt keine Komplikationen beim Download, beim Setup und bei den Konfigurationsschritten, und Sie arbeiten garantiert immer mit der neuesten Version.

Für vollwertige Anwendungen bietet Java Web Start dieselben Vorteile wie für die oben beschriebenen HTML-basierten Anwendungen. Java Web Start ist eine Lösung für den Einsatz von Anwendungen über das Internet. Die Verwendung von vollwertigen Anwendungen anstelle von HTML-basierten Clients kann viel Vorteile bringen:

- Eine in hohem Maße interaktive Benutzeroberfläche, die mit herkömmlichen Anwendungen wie Textverarbeitung und Tabellenkalkulation vergleichbar ist.
- Geringere Bandbreitenanforderungen. Eine Anwendung muss nicht unbedingt bei jedem Klick Daten mit dem Web-Server austauschen und kann bereits heruntergeladene Informationen zwischenspeichern. Dadurch kann bei langsamen Verbindungen eine bessere Interaktivität gewährleistet werden.
- Möglichkeit des Offline-Betriebs.

Als Kompromiss muss die Anwendung beim ersten Mal aus dem Internet geladen werden. Mit HTML-basierten Anwendungen sind sehr niedrige Anforderungen für die *Erstaktivierung* verbunden. In der Regel ist eine Webseite innerhalb weniger

Sekunden aufgebaut. Für eine Anwendung, die auf der Java™-Technologie basiert, sind mit einer üblichen Modemverbindung Download-Zeiten in der Größenordnung von mehreren Minuten erforderlich. Java Web Start speichert alle heruntergeladenen Dateien lokal auf dem Computer. Obwohl die Anforderungen für die Erstaktivierung bei Anwendungen höher sind als bei HTML-Seiten, erfolgt der Start der Anwendung bei den nachfolgenden Starts fast sofort, da alle erforderlichen Ressourcen bereits lokal verfügbar sind.

Bei jedem Start fragt Java Web Start beim Web-Server an, ob eine neue Version der Anwendung verfügbar ist und lädt und startet diese automatisch. So werden Anwendungen automatisch aktualisiert. Es gibt keinen komplizierten Aktualisierungsablauf.

1.7.2. Sicherheit

Java Web Start setzt auf der Java 2-Plattform und ihrer umfassenden Sicherheitsarchitektur auf. Mit Java Web Start gestartete Anwendungen werden standardmäßig in einer eingeschränkten Umgebung ausgeführt ("Sandbox"), in der der Zugriff auf Dateien und das Netzwerk beschränkt ist. Beim Start von Anwendungen mit Java Web Start bleiben also Systemsicherheit und -integrität erhalten.

Eine Anwendung kann den unbeschränkten Zugriff auf Ihr System anfordern. In diesem Fall zeigt Java Web Start beim ersten Start der Anwendung ein Dialogfeld *Sicherheitswarnung* an. Die Sicherheitswarnung enthält Informationen zum Ursprung, also den Hersteller, der die Anwendung entwickelt hat. Wenn Sie dem Hersteller vertrauen, wird die Anwendung gestartet. Die Informationen über den Ursprung der Anwendung basiert auf der Signatur mit einem digitalen Code.

1.7.3. Verwenden von Java Web Start

Mit Java Web Start können Anwendungen, die auf der Java-Technologie basieren, direkt über das Internet gestartet werden. Eine Anwendung kann auf drei verschiedene Arten gestartet werden:

- Durch Klicken auf einen Link in einem Web-Browser.
- Aus dem in Java Web Start integrierten Anwendungsmanager heraus, der in letzter Zeit verwendete Anwendungen aufzeichnet und den schnellen Zugriff auf Ihre Favoriten ermöglicht.
- Über Desktop-Symbole oder aus dem Startmenü (nur bei Windows).

Auf die Ausführung der Anwendung selbst hat die Startmethode keinen Einfluss. Insbesondere fragt Java Web Start immer beim Web-Server an, um festzustellen, ob eine aktuellere Version der jeweiligen Anwendung vorliegt.

1.7.3.1. Start über einen Web-Browser

Auf der Produktwebseite von Java Web Start (http://java.sun.com/products/javawebstart) befinden sich Links zu einer Reihe von Anwendungen, die mit einem einzigen Mausklick gestartet werden können. Probieren Sie die verschiedenen Anwendungen aus, indem Sie auf die *Start*-Schaltflächen klicken. Daraufhin startet der Web-Browser Java Web Start, das dann die jeweilige

Anwendung herunterlädt, im Speicher hält und ausführt. Sie werden feststellen, dass die Anwendung beim zweiten Start viel schneller bereit ist, da sie bereits lokal gespeichert ist und nicht nochmals heruntergeladen werden muss.

Die meisten Demonstrationsprogramme auf der Webseite werden ohne Benutzereingriff heruntergeladen und ausgeführt. Diese Anwendungen werden in einer beschränkten Umgebung ausgeführt und haben keinen Zugriff auf die lokale Festplatte und das Netzwerk. Sie können garantiert keinen Virus auf Ihrer Festplatte installieren.

Einige der Demonstrationsprogramme erfordern zusätzliche Privilegien, z. B. den Zugriff auf Ihre lokale Festplatte. Bei diesen Anwendungen wird ein Sicherheitsdialogfeld mit Informationen über die Herkunft der Anwendung angezeigt, wobei entscheidend ist, wer den Code digital signiert hat. Die Anwendung wird nur ausgeführt, wenn Sie dem Hersteller vertrauen.

Das war schon alles zur Verwendung von Java Web Start, doch wie funktioniert Java Web Start eigentlich? Die HTML-Links, über die die Anwendungen gestartet werden, sind ganz normale HTML-Links. Statt auf eine andere Webseite zu zeigen, stellen sie jedoch die Verbindung mit einer speziellen Konfigurationsdatei her, einer so genannten JNLP-Datei. Der Web-Browser analysiert die Dateierweiterung und/oder den MIME-Typ der Datei und ordnet die Datei Java Web Start zu. Anschließend startet der Web-Browser Java Web Start und übergibt die heruntergeladene JNLP-Datei als Argument. Java Web Start zeigt einen Startbildschirm an und lädt die Anwendung, hält sie im Speicher und führt sie aus, wie in der JNLP-Datei angegeben.

1.7.3.2. Start aus dem integrierten Anwendungsmanager heraus

Der Anwendungsmanager ist ein integraler Bestandteil von Java Web Start. Damit können Anwendungen schnell und einfach gestartet werden, die vorher von Java Web Start gestartet wurden. Er besteht aus einer Kombination eines Verlaufsmenüs und eines Start-/Programmmenüs für Ihre über das Internet bereitgestellten, auf der Java-Technologie basierenden Anwendungen. Über den Anwendungsmanager können auch zusätzliche Informationen über eine Anwendung angezeigt werden, und Sie können die Homepage einer Anwendung öffnen.

Eine Anwendung kann im Anwendungsmanager durch Doppelklick auf das Symbol der Anwendung oder durch Klicken auf die Schaltfläche *Start* gestartet werden.

Eine weitere wichtige Funktion des Anwendungsmanagers ist das Dialogfeld Einstellungen, mit dem die von Java Web Start verwendeten Einstellungen kontrolliert und angepasst werden können. So stehen zum Beispiel Register für folgende Aufgaben zur Verfügung:

- Angabe eines HTTP-Proxys (oder Java Web Start kann angewiesen werden, die standardmäßigen Browser-Einstellungen zu verwenden).
- Löschen von heruntergeladenen Anwendungen aus dem Cache.
- Angabe der Verzeichnisse der verschiedenen Versionen von Java Runtime Environments (JREs).
- Auswahl, ob eine Java-Console angezeigt werden soll.

- Anzeigen der Root-Zertifikate.

Der Anwendungsmanager wird entweder durch Klicken auf das Symbol *Java Web Start* auf dem Desktop oder im Startmenü von Windows gestartet. Unter Solaris/Linux erfolgt der Start über den Aufruf des Befehls `javaws` im Installationsverzeichnis von Java Web Start. Der Anwendungsmanager kann auch von einem Web-Browser aus gestartet werden. Auf der Produkt-Webseite (http://java.sun.com/products/javawebstart) ist ein Beispiel dazu gezeigt.

1.7.3.3. Start über Desktop-Symbole bzw. über das Startmenü (nur unter Windows)

Java Web Start kann automatisch Verknüpfungen für über das Internet bereitgestellte, auf der Java-Technologie basierende Anwendungen auf dem Windows-Desktop und im Startmenü erstellen. Standardmäßig fragt Java Web Start beim zweiten Start einer Anwendung, ob eine Verknüpfung erstellt werden soll. Über das Einstellungspanel kann dies geändert werden.

Verknüpfungen können auch mit dem Anwendungsmanager hinzugefügt und entfernt werden (über die Menüpunkte *Anwendung/Verknüpfung erstellen* und *Anwendung/Verknüpfung entfernen*).

1.7.3.4. Verwenden von Java Web Start hinter einem Proxy-Server bzw. einer Firewall

Um Anwendungen außerhalb Ihrer Firewall starten zu können, muss Java Web Start mit den richtigen Proxy-Einstellungen konfiguriert werden. Java Web Start versucht automatisch, die Proxy-Einstellungen Ihres Standard-Browsers im System zu erkennen (IE oder Netscape unter Windows und Netscape unter Solaris/Linux). Von Java Web Start werden die meisten Autokonfigurations-Scripts auf Web-Proxys unterstützt. Die Proxy-Einstellungen werden in nahezu allen Umgebungen erkannt.

Wenn die Proxy-Einstellungen nicht automatisch erkannt werden können, werden Sie bei der ersten Verwendung von Java Web Start aufgefordert, die Proxy-Einstellungen anzugeben. Außerdem werden Sie von Java Web Start aufgefordert, den für den Zugriff auf einen Proxy-Server mit Authentifizierung erforderlichen Benutzernamen und das Passwort anzugeben. Der Benutzername und das Passwort werden für den aktuellen Aufruf von Java Web Start gespeichert. Beim Zugriff auf eine sichere Website werden der Benutzername und das Passwort jedoch von einer neu gestarteten Java Virtual Machine abgefragt, da diese Informationen in der Instanz einer Java Virtual Machine gespeichert werden.

Die Proxy-Konfiguration kann auch mit dem Einstellungspanel von Java Web Start angezeigt bzw. verändert werden. Starten Sie den Anwendungsmanager entweder durch Klicken auf das Symbol auf dem Desktop (Windows) oder durch Eingabe von `./javaws` im Installationsverzeichnis von Java Web Start (Solaris/Linux), und wählen Sie dann Bearbeiten/Einstellungen. Wenn Sie sich in einer Umgebung befinden, in der der Zugang zum Internet über einen Proxy-Server erfolgt, ist es ratsam, mit dem Einstellungspanel von Java Web Start zu prüfen, ob die Einstellungen richtig sind.