

Mit IzPack ein Installationsprogramm erstellen

Schön verpackt

■ VON THORSTEN KAMANN

Sie haben eine Anwendung entwickelt und alle Tests wurden erfolgreich absolviert. Jetzt stellt sich die Frage, wie man die Anwendung am besten verteilt. Eine Möglichkeit ist es, die Binaries einfach zu zippen und das Archiv zum Download anzubieten. Dies ist oftmals nicht ausreichend. Also muss ein Installationsprogramm her. Dieser Workshop führt Sie durch die notwendigen Schritte, um ein solches Programm zu erstellen, das kommerziellen Lösungen in nichts nachsteht.

Um ein Installationsprogramm (im Folgenden Installer genannt) zu erstellen, werden Sie immer eine Art Generator brauchen, der aus Ihren Eingaben und Konfigurationen den entsprechenden Installer erstellt. IzPack [1] ist ein solcher Generator. Die Konfiguration wird dort in einer XML-Datei gespeichert und der IzPack Compiler generiert ein ausführbares JAR-Archiv.

Ein Installer besteht gewöhnlich aus einer Menge Screens, die über Buttons durchgeklickt werden können. IzPack bietet einen Rahmen, der die Navigation (WEITER | ZURÜCK | ABBRECHEN) zur Verfügung stellt. Die Screens werden mittels Panels bereitgestellt, die über die XML-Datei konfiguriert werden. Es werden für die meisten Anwendungsfälle schon fertige Panels angeboten, es ist aber problemlos möglich, eigene zu integrieren. Weitere Informationen zu IzPack finden Sie im Kasten „Facts zu IzPack“.

Dieser Artikel ist als Workshop konzipiert. Aus diesem Grund finden Sie auf der Heft-CD ein Beispielprojekt, das einen Installer für den Tomcat-Servlet-Container erstellt. Um die einzelnen Schritte und Erklärungen nachvollziehen zu können, ist es sinnvoll, wenn Sie parallel zu diesem Artikel das Beispielprojekt in Eclipse öffnen. Im Kasten „Inhalt des Workshops“ finden Sie detaillierte Informationen über den Funk-

tionsumfang des Installers, den Sie mithilfe dieses Workshops erstellen. Sie müssen nur zu Beginn die Eigenschaft `izpack.home.dir` in der Datei `project.properties` auf das Installationsverzeichnis von IzPack (ebenefalls auf der Heft-CD) ändern.

Wichtige Konzepte von IzPack

IzPack basiert auf einigen wenigen Konzepten, die das Arbeiten mit dem Generator vereinfachen.

- **Variablenersetzung:** Eingabewerte in Formularen werden in Variablen gespeichert. Diese können in Dateien au-

tomatisch ersetzt werden, wenn diese als *parsable* gekennzeichnet werden. Variablen werden mit dem Präfix \$ gekennzeichnet. Der Name einer Variablen darf keine Punkte enthalten, benutzen Sie „_“ als Trennung.

- **Ressourcen:** Viele Panels brauchen externe Ressourcen. Das können Bilder, Sprachtexte, Konfigurationsdateien oder Ähnliches sein. Das Panel hat die IDs der Ressourcen fest kodiert, innerhalb des Ressourcenelements können Sie die Quelle definieren:

Facts zu IzPack

IzPack ist ein Open-Source-Projekt, welches 2001 von Julien Ponge gegründet wurde und inzwischen zu den aktivsten Projekten auf BerliOS [5] gehört. Die Hauptfeatures sind:

- Open Source; ist unter einer Apache-Lizenz veröffentlicht
- unterstützt eine Vielzahl von Sprachen (z.Z. mehr als 20)
- XML-basiert, dadurch modular und erweiterbar
- kann sehr leicht in Ant integriert werden
- kann Ant einfach in den Installationsprozess integrieren
- nicht abhängig vom nativen Code, kann aber native Bibliotheken benutzen
- kann Benutzereingaben entgegennehmen und validieren
- unterstützt Replacements, externe Skripte und die Systemumgebung
- kann Shortcuts in Windows und FreeDesktop.org-kompatible X11-Server erstellen

Inhalt des Workshops

Das Ziel des Workshops ist es, ein Installationsprogramm zu erstellen, mit dem Tomcat installiert werden kann. Als Mehrwert gegenüber dem Windows-Installer oder dem ZIP-/TAR.GZ-Archiv soll es möglich sein, folgende Einstellungen vorzunehmen:

- Installationsverzeichnis frei wählbar
- zusätzliche Packages für die Admin-Webapp und den Deployer optional installierbar
- Serverport
- HTTP-Port
- HTTPS-Port konfigurierbar
- Xms und Xmx-Werte frei konfigurierbar
- Benutzer-Account für den Administrator anlegbar

Um das Ziel zu erreichen, werde ich Sie Schritt für Schritt durch die Erstellung der notwendigen Dateien führen und Ihnen auch zeigen, an welchen Stellen Sie in die Konfigurationsdateien des Tomcat eingreifen müssen. Am Ende des Workshops haben Sie einen Installer, mit dem Sie Tomcat auf Windows, Linux, Mac mit verschiedenen Konfigurationen installieren können.



Quellcode auf CD

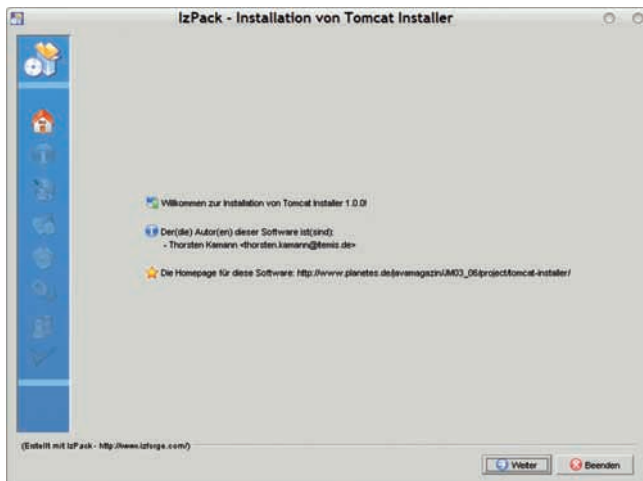


Abb. 1: Willkommen-Screen

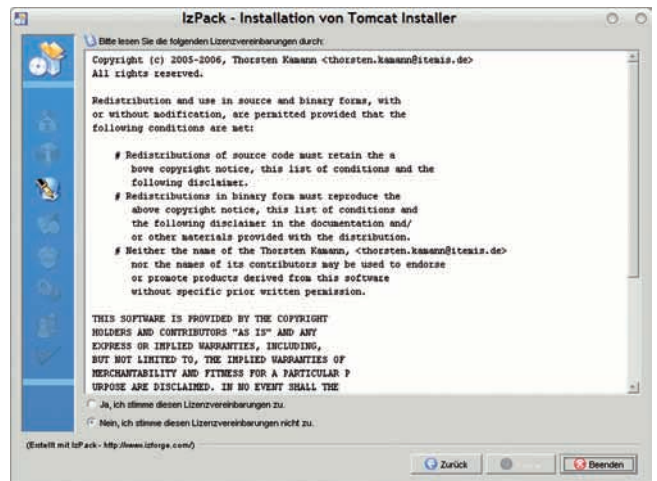


Abb. 2: Lizenz-Screen

```
<resources>
  <res id="RESOURCES.ID" src="RESOURCES.SRC"/>
  [...]
</resources>
```

- **Internationalisierung:** Der zu erstellende Installer kann vollständig internationalisiert werden. IzPack bietet bereits Sprachdateien für 24 Sprachen. Die mitgelieferten Sprachdateien sind allerdings nur für die Standard-Panels. Wenn Sie z.B. ein `UserInputPanel` benutzen, müssen Sie die Texte für die Formulare natürlich selbst internationalisieren.
- **Panels:** Ein Panel basiert auf `javax.swing.JPanel` und kann somit einfach selbst erstellt werden. Für die meisten Anwendungsfälle existieren bereits fertige Panels. Insbesondere das `UserInputPanel`, mit dem Sie sehr flexible Formulare gestalten können, wird die meisten Bedürfnisse abdecken.

- **Packs:** Packs beschreiben ein Installationspaket. Ein solches Paket kann aus beliebigen Dateiressourcen bestehen, die mit File Sets konfiguriert werden können. Zusätzlich können einzelne Dateien als *parsable* oder als *executable* markiert werden. Ein Pack kann abhängig von anderen Packs und/oder als *optional* gekennzeichnet werden.
- **Eigene Bibliotheken:** Sie können auch eigene Bibliotheken in den Installer einfügen, die die Funktionalität des Installers erweitern. Dies können JAR-Archive und native Bibliotheken sein.

installer.xml. Im selben Verzeichnis befindet sich auch eine DTD, die das Format dieser XML beschreibt, allerdings hat der interne XML-Parser Probleme beim Auflösen der DTD, aus diesem Grund hat die XML-Datei keinen Verweis auf die DTD. Ein Installer hat typischerweise mehrere Screens, um den Benutzer durch den Installationsprozess zu führen. In IzPack werden diese Screens mittels der Panels realisiert.

Der Willkommen-Screen

Der erste Screen, den der Anwender zu sehen bekommt, ist eine Begrüßung, in der einige Informationen über den Installer und die Autoren stehen (Abb. 1). Dafür gibt es in IzPack das `HelloPanel`. Dieses Panel wird im `<panels>`-Bereich konfiguriert: `<panel classname="HelloPanel"/>`. Die Informationen, die auf diesem Panel angezeigt werden, befinden sich in der `tomcat-installer.xml` im `<info>`-Bereich (Listing 1).

Vorbereitung der Installationsdateien

Die zu installierenden Dateien liegen am besten im Projekt in einem Verzeichnis `packs`. Dort können Sie die Packs beliebig benennen. Im Beispielprojekt heißen sie `core`, `admin` und `deployer`. Damit die vom Anwender eingegebenen Daten auch in den entsprechenden Konfigurationsdateien des Tomcat ersetzt werden, werden einige Dateien angepasst. Das sind

Der Lizenz-Screen

Der nächste Screen zeigt die Lizenz, die der Anwender akzeptieren muss (Abb. 2). Auch dieses Panel wird im `<panel>`-Bereich konfiguriert: `<panel classname="LicencePanel"/>`. Der Text der Lizenz kommt aus einer Ressource. Dies ist gewöhnlich eine Textdatei:

```
<res id="LicencePanel.licence" src="resources/licence.txt"/>
```

- `core/bin/catalina.bat`
- `core/bin/catalina.sh`
- `core/conf/server.xml`
- `core/conf/tomcat-users.xml`

In diesen Dateien finden Sie die IzPack-Variablen, die Sie später im `UserInputPanel` definieren.

Konfiguration von IzPack

Die Konfiguration für den Generator von IzPack befindet sich in `installer/tomcat-`

Wenn Sie anstelle von Plain-Text lieber eine HTML-Datei benutzen wollen, dann benutzen Sie als Panel das `HTMLLicence-`

Listing 1

Informationen für das WelcomePanel

```
<info>
<appname>Tomcat Installer</appname>
<appversion>1.0.0</appversion>
<authors>
  <author email="thorsten.kamann@itemis.de"
    name="Thorsten Kamann" />
</authors>
<url>http://www.planetes.de/javamagazin/JM03_06/
  project/tomcat-installer</url>
<uninstaller name="tomcat-uninstaller.jar"
  write="yes" />
</info>
```

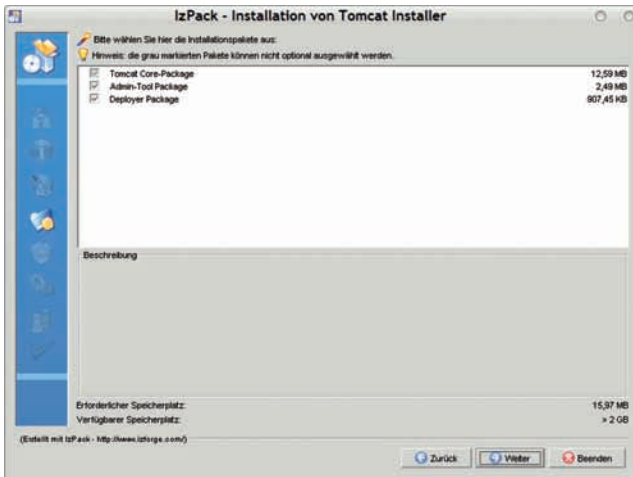


Abb. 3: Packs-Screen

Panel. Die Ressource heißt dann dementsprechend *HTMLLicencePanel.licence*.

Der Package-Auswahl-Screen

Wie weiter oben bereits beschrieben, gibt es zwei optionale Pakete, die Sie mit installieren können. Im Package-Auswahl-Screen haben Sie die Möglichkeit dazu (Abb. 3). Für diesen Screen benutzen Sie das *PacksPanel*: `<panelclassname="PacksPanel"/>`. Die einzelnen Packs konfigurieren Sie über das `<pack>`-Element (Listing 2). Eine genaue Dokumentation über dieses Element finden Sie in der IzPack-Dokumentation im Beispielprojekt unter *IzPack/doc/izpack/xhtml/index.html*.

Der Zielverzeichnis-Screen

Der nächste Screen fordert den Anwender auf, ein Zielverzeichnis zu wählen. Sie können mittels Ressourcen für die verschiedenen Plattformen (Windows, Unix, Mac) bereits einen Vorschlag definieren. Für diesen Screen (Abb. 4) benutzen Sie das *TargetPanel*: `<panelclassname="TargetPanel"/>`. Für die einzelnen Plattformen können Sie unterschiedliche Vorschlagswerte bestimmen. Der folgende Code beschreibt das Vorgehen mit Ressourcen:

```
<res id="TargetPanel.dir.unix" src="resources/target/
target.unix.res"/>
<res id="TargetPanel.dir.windows" src="resources/target/
target.windows.res"/>
<res id="TargetPanel.dir.mac" src="resources/target/
target.mac.res"/>
```

Innerhalb der einzelnen Ressourcen-Dateien brauchen Sie nur eine Zeile eintra-

gen, in der der Pfad zu dem vorgeschlagenen Installationsverzeichnis steht.

Der Einstellungen-Screen

Damit der Anwender einige Einstellungen wie Port und Administrations-Account

Listing 2

Definition der einzelnen Packs

```
<packs>
<pack name="Tomcat Core-Package" required="yes">
<description>The files for the Tomcat Core-Package</
description>
<fileset dir="packs/core" targetdir="$INSTALL_PATH">
<include name="**"*/>
</fileset>
<parsable targetfile="$INSTALL_PATH/conf/server.
xml" type="xml"/>
<parsable targetfile="$INSTALL_PATH/conf/
tomcat-users.xml" type="javaprop"/>
<parsable targetfile="$INSTALL_PATH/bin/catalina.
bat" type="plain"/>
<parsable targetfile="$INSTALL_PATH/bin/catalina.
sh" type="plain"/>
</pack>
<pack name="Admin-Tool Package" required="no">
<description>The files for the Admin-Webapplication</
description>
<depends packname="Tomcat Core-Package"/>
<fileset dir="packs/admin" targetdir="$INSTALL_PATH">
<include name="**"*/>
</fileset>
</pack>
<pack name="Deployer Package" required="no">
<description>The files needed for remote deployment
of webapps</description>
<depends packname="Tomcat Core-Package"/>
<fileset dir="packs/deployer" targetdir="$INSTALL_PATH/deployer">
<include name="**"*/>
</fileset>
</pack>
</packs>
```

Anzeige

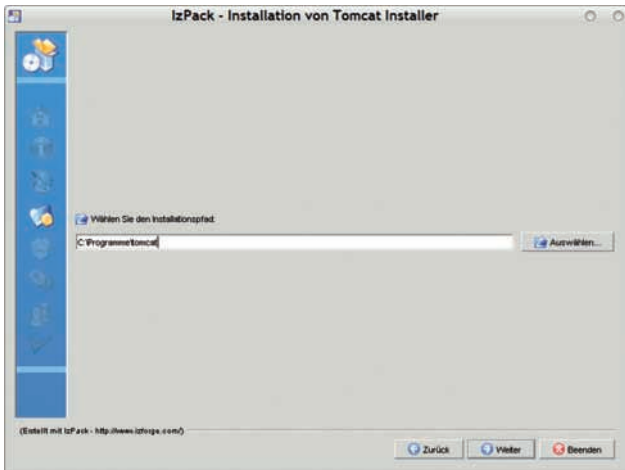


Abb. 4: Auswahl des Zielverzeichnis

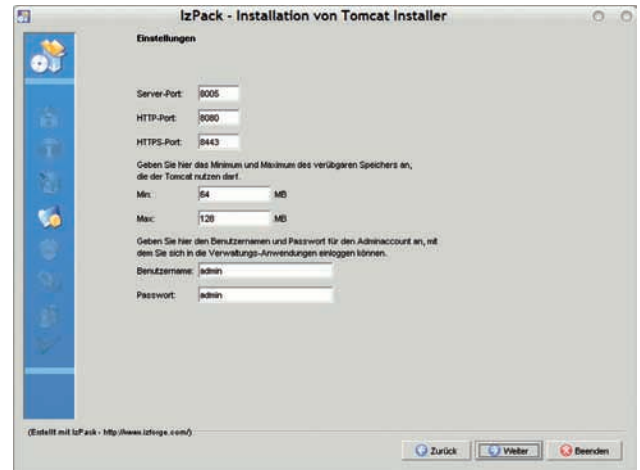


Abb. 5: Benutzereinstellungen

an seine Bedürfnisse anpassen kann, ist noch ein Screen nötig, in dem ein Formular mit den nötigen Eingabefeldern angezeigt wird (Abb. 5). Dies kann man mit dem *UserInputPanel* realisieren. Dieses Panel ist das mächtigste und daher leider das komplizierteste Panel. Die Definition

erfolgt auf dem üblichen Weg: `<panel class name="UserInputPanel"/>`.

Die Konfiguration des Eingabefelders wird mittels *Resource* auf eine externe Datei verlagert. Es ist notwendig, die Labels und Meldungen anhand einer Sprachdatei zu definieren:

```
<res id="userInputSpec.xml" src="resources/input/input.xml"/>
<res id="userInputLang.xml_deu" src="resources/i18n/deu.xml"/>
```

In dem Formular werden fast ausschließlich *RuleInputFields* verwendet, die Prozessoren und Validatoren ermöglichen. Darüber hinaus kann man Feldinhalte und -präsentation mit Eingabe-Strings konfigurieren. Validatoren überprüfen den Feldinhalt beim Klick auf den WEITER-Button, während Prozessoren bei der Erstellung – also beim Start des ge-

nerierten Installers – ausgeführt werden. In dem Beispielprojekt wird der Built-in *PortProcessor* verwendet. Aus der Zeile `set="0:8080:com.izforge.izpack.util.PortProcessor"` ermittelt der *PortProcessor*, ob der Standard-Port 8080 in Verwendung ist. Ist er das, dann wird die Portnummer solange um 1 erhöht, bis ein freier Port gefunden wurde. Dieser wird dann in das Feld eingetragen. Der Validator überprüft dagegen, ob der Feldinhalt korrekt ist. Der – ebenfalls Built-in – *PortValidator* überprüft, ob der eingegebene Port in Verwendung ist oder nicht. Eine Fehlermeldung wird angezeigt, wenn er belegt ist. Beide Konfigurationen finden Sie im Beispielprojekt.

Der Installations-Screen

Der nächste Screen, den der Anwender zu sehen bekommt, ist der Installations-Screen (Abb. 6). Dort werden die Dateien

Weitere Features von IzPack

Es gibt einige Features von IzPack, die das Beispielprojekt nicht verwendet hat und die somit in diesem Workshop keine Beachtung gefunden haben.

- Es gibt noch eine Anzahl weiterer Panels:
 - *ImgPacksPanel*: Die einzelnen Packs können mit Bildern illustriert werden.
 - *XinfoPanel*: Die Platzhalter im Text dieses Panel werden mit den Variablenwerten ersetzt.
 - *ShortcutPanel*: erzeugt konfigurierbare Shortcuts.
 - *CompilePanel*: kompiliert (generierte) Java-Quellen zur Laufzeit.
 - *JDKPathPanel*: sucht nach einem konfigurierten JRE/JDK.
- Systemeigenschaften sind als Variablen verfügbar (`$SYSTEM_SYSTEMVARIABLE`). Punkte im Namen der Systemvariablen gegen „_“ ersetzen.
- Packages können auch aus dem Internet nachgeladen werden (WebInstaller). Dazu fügen Sie einfach in den `<info>`-Bereich `<webdir>http://www.superextractor.com/download</url>` ein.
- Shortcuts können beliebig konfiguriert werden. Das funktioniert sowohl bei Windows als auch KDE/Gnome einwandfrei.
- *CustomActions* sind Listeners, die sich in die verschiedensten Stellen des Installationsprozesses einklinken können.

Listing 3

Definition der Prozesse

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<processing>
  <logfiledir>$INSTALL_PATH/logs</logfiledir>
  <job name="Starting Tomcat">
    <os family="unix" />
    <executable name="$INSTALL_PATH/bin/catalina.sh">
      <arg>start</arg>
    </executable>
  </job>
  <job name="Starting Tomcat">
    <os family="windows" />
    <executable name="$INSTALL_PATH/bin/catalina.bat">
      <arg>start</arg>
    </executable>
  </job>
  <job name="Starting Tomcat">
    <os family="mac" />
  </processing>
```

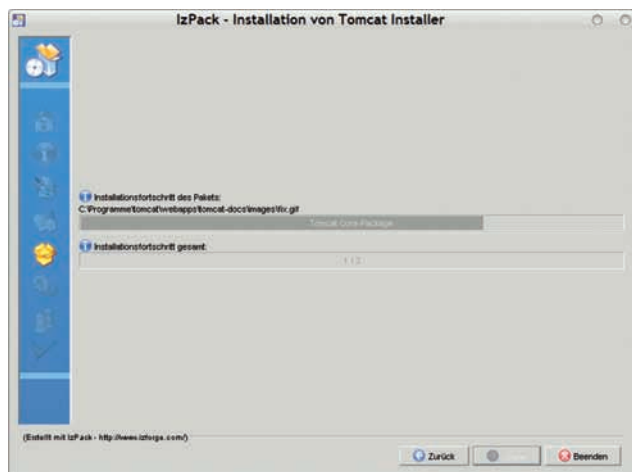


Abb. 6: Kopieren der Dateien der ausgewählten Packs

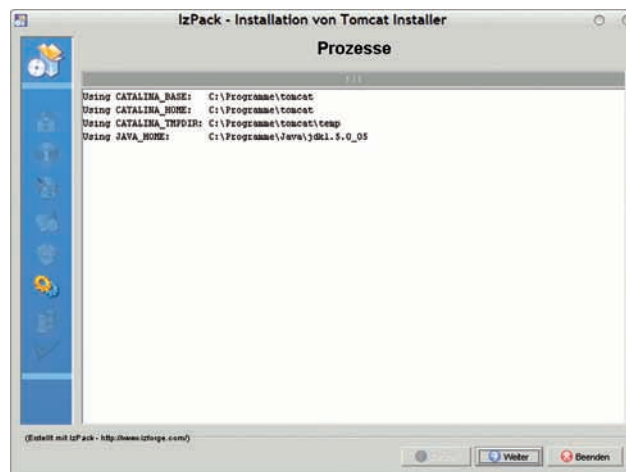


Abb. 7: Ausführung der Prozesse

der ausgewählten Packs kopiert, alle Platzhalter der *parsable*-Dateien ersetzt und die Dateien ausgeführt, die in der Packs-Konfiguration als *executable* und *stage="postinstall"* gekennzeichnet sind. Dieser Screen wird mit dem *InstallPanel* ausgeführt:

```
<panel classname="Install-Panel"/>
```

Konfiguration der PostInstall-Prozesse

Nachdem alle Dateien kopiert wurden, wäre es schön, wenn der Tomcat automatisch gestartet würde, wie in Abbildung 7 zu sehen. IzPack bietet die Möglichkeit, zu einem beliebigen Zeitpunkt Prozesse auszuführen. Dies können native Programme, aber auch Java-Klassen und Ant-Skripte sein. Das führt das *ProcessPanel* aus:

```
<panel classname="ProcessPanel"/>
```

Die Definition der Prozesse wird in einer externen Ressource definiert:

```
<res id="ProcessPanel.Spec.xml" src="resources/process/process.xml"/>
```

In der Prozesskonfiguration können Sie beliebig viele Prozesse definieren, die nacheinander ausgeführt werden sollen. In unserem Fall gibt es nur einen, der aber für jede Plattform (Windows, Unix, Mac) unterschiedlich ist (Listing 3).

Der Finish-Screen sowie Installer generieren und ausführen

Der letzte Screen, den der Anwender sieht, ist ein FinishScreen. Diese Aufgabe übernimmt das *SimpleFinishPanel*:

```
<panel classname="SimpleFinishPanel"/>
```

Im Projektverzeichnis generiert eine *build.xml* den Installer. Der Installer

selbst befindet sich danach im Verzeichnis *dist*. Dort können Sie ihn mit einem Doppelklick direkt ausführen.

Resümee

Es ist möglich, mit IzPack schnell und einfach einen ansprechenden Installer zu generieren, der auf allen Plattformen funktioniert, für die es eine JVM und ein GUI gibt. Das Aussehen hat natürlich einen Swing-typischen Touch und sieht auf dem einen oder anderen Betriebssystem nicht so gut aus. Doch kann man durch die Looks-Integration vieles ausmerzen. Für eine richtig native Integration ist evtl. ein SWT-Port denkbar.

Den Installer aus diesem Workshop kann man ohne weiteres in ein bis zwei Stunden fertigstellen. Er bietet noch viele Erweiterungsmöglichkeiten. Z.B. wird kein Manager Account angelegt, es fehlt eine Integration eines Service Wrapper und des JK Connectors. All dies lässt sich ohne einen Eingriff in IzPack realisieren.

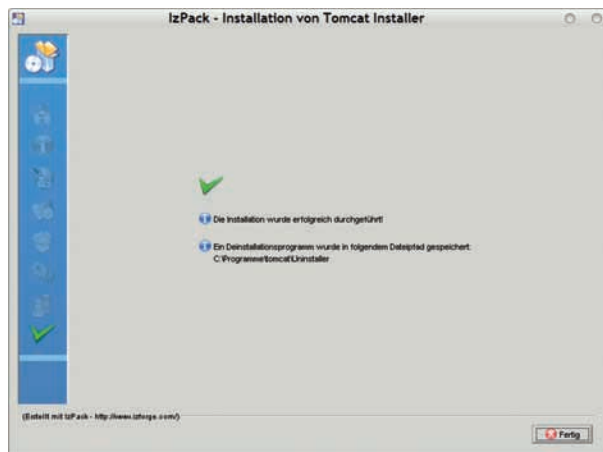


Abb. 8: Der Finish-Screen



Thorsten Kamann (thorsten.kamann@itemis.de) ist Softwareentwickler und IT-Consultant bei Itemis. Dort arbeitet er in Projekten, die als Schwerpunkt Java EE-Anwendungen mit MDA-Architekturen haben.

Links & Literatur

- [1] www.izforge.com/izpack/
- [2] developer.berlios.de/mail/?group_id=1408
- [3] developer.berlios.de/bugs/?group_id=1408
- [4] www.izforge.com/izpack/izpack-doc.pdf
- [5] www.berlios.de
- [6] tomcat.apache.org/download-55.cgi
- [7] www.jgoodies.com/freeware/looks/