

Plus CD!

Xtext: Erweiterung und Ausführung von DSLs

6.09

Deutschland € 9,80
Österreich € 10,80, Schweiz sFr 19,20



eclipse
MAGAZIN

eclipse

MAGAZIN

www.eclipse-magazin.de

EXKLUSIV AUF CD:

Xpert.ivy

Die Business Process Management
Plattform

Apache Directory Studio 1.4.0

Open Source LDAP-Browser und Directory
Client

Java Workflow Tooling

Werkzeuge zur Ausführung von
Geschäftsprozessen

MiGLayout

Layout-Manager für SWT
und Swing

jetty @ eclipse

► **POWER FÜR ECLIPSE RUNTIME**

Was der Jetty-Webserver bei Eclipse bewirken kann

► **MIT JETTY IN DIE WOLKE**

Greg Wilkins über die Intalio-Übernahme

► **JETTY-TUTORIAL**

Eine Jetty-getriebene Webanwendung bauen

w-jax09

Alle Infos zur Konferenz
siehe Beihefter

Datenträger enthält
Info- und
Lehrprogramme
gemäß §14 JuSchG

► **Modellierung in 3D** >> 60
Dreidimensionale Editoren erstellen
mit GEF3D

► **Zest Visualization
Toolkit** >> 86
Darstellung von Grafen leicht gemacht

► **Eclipse Memory
Analyzer** >> 46
Auf der Suche nach den
Speicherlecks

► **Hot-Plugging mit RCP
und OSGi** >> 33
Anwendungen im laufenden Betrieb
anpassen

D 68864



4 196603 209806 0 6



RCP Usability: Wie erleichtere ich meinen Anwendern den Einstieg?

Die ersten Schritte sind die schwersten

Quellcode
auf CD!

>> ALEXANDRA IMRIE UND ACHIM LÖRKE

Benutzerfreundlichkeit ist ein wichtiger Teil der Anwendungsentwicklung. Ein überlegtes Benutzerkonzept und eine einfache Bedienung werden jedoch häufig in der Planung nicht weitgehend betrachtet. Oft wird Benutzerfreundlichkeit nur nachgerüstet, nachdem Kunden schon mit Problemen zu kämpfen hatten. Sinnvoller ist es, Benutzerfreundlichkeit früh einzubeziehen, damit eine Software schon von Anfang an einen guten Eindruck hinterlässt. Die Eclipse-Rich-Client-Plattform bietet mehrere Möglichkeiten an, die Benutzerfreundlichkeit einer Anwendung zu verbessern. In diesem Artikel werden zwei dieser Optionen erläutert.

Die Vorteile, frühzeitig über Benutzerfreundlichkeit nachzudenken, sind zahlreich: Zum einen trägt sie zum wichtigen ersten Eindruck einer Software bei.

Ein weiterer Vorteil ist die Kundenzufriedenheit, wenn das Produkt auch ohne die 1 000 Seiten Dokumentation verständlich ist. Auch finanzielle Gründe spielen

eine Rolle, denn es ist aufwändiger, Benutzerfreundlichkeit später hinzuzufügen, als sie von Anfang an einzuplanen. Man spart zudem Geld in der Gewährleistung und im Support, weil der Kunde nicht ständig wegen Kleinigkeiten nachfragen muss. Dabei ist es nicht schwierig, eine Benutzerhilfe in sein Produkt einzubauen. Dieser Artikel beschäftigt sich mit zwei Eclipse-Konstrukten: „Welcome Pages“ und „Cheat Sheets“. Deren Implementierung ist kein Ersatz für ein wohlüberlegtes Bedienkonzept, kann aber die notwendige Hilfestellung bieten, um dem Benutzer den Einstieg zu erleichtern.

Die Beispielanwendung

Unser wichtigstes Anliegen in diesem Artikel ist es zu zeigen, wie einfach



die Grundlagen für Benutzerhilfen zu schaffen sind, und auf die kleinen Fallen hinzuweisen, die bei der Umsetzung lauern. Als Basis für unsere Beispiele dient eine altbekannte RCP-Anwendung: das Mail-Plug-in. Diese Anwendung lässt sich innerhalb der IDE durch FILE | NEW>PROJECT... erstellen. Um eine identische Basis für alle Beispiele zu haben, sollte man sich alle Projekte aus dem bereitgestellten Archiv (siehe Kasten) in einen Workspace installieren. Die generierte RCP-Mail-Anwendung steht im Projekt *base* bereit.

Wir gehen auch davon aus, dass Plug-ins und RCP-Anwendungen für Sie kein Neuland sind, daher wird hier weitgehend auf Schritt-für-Schritt-Erklärungen verzichtet. Alle Umsetzungen sind unsere „Best Practices“, d. h. sie wurden bereits in Produkten eingesetzt und funktionieren problemlos. Sie haben allerdings beileibe keinen Anspruch auf Vollständigkeit oder Ausschließlichkeit. Für sehr komplexe Systeme sind sie sicher nicht hinreichend, aber sie bilden einen guten Ausgangspunkt.

Welcome Pages

Bei vielen Anwendungen gibt es wichtige Informationen, die ein Benutzer sehen sollte, bevor er beginnt zu arbeiten. Selbst kurze PDF-Einleitungen sind aber kein beliebtes Lesematerial. Man braucht idealerweise eine Informationsquelle, die direkt in die Anwendung eingebaut ist, um dem Anwender wichtige Informationen und Neuigkeiten mitzuteilen. Die Welcome Pages erfüllen diesen Bedarf, da sie beim Neustart der Anwendung erscheinen. Das HTML-Format wirkt freundlich und bietet anhand von Grafiken und kurzen Texten eine Stelle, um den Benutzer über Neuigkeiten, erste Schritte, Webressourcen, Beispiele und Migration zu informieren. Beim Schreiben der HTML-Seiten sollte man beachten, dass man die Dokumentation nicht einfach wiedergibt, sondern sich auf das Wesentliche konzentriert. Sinnvolle Verlinkungen zwischen Seiten machen auch die Welcome Pages benutzerfreundlicher.

Die Welcome Pages können komplett ohne Programmierung erstellt werden. Zu ihrer Nutzung müssen jedoch einige Plug-ins bereitgestellt werden. Dies kann entweder dadurch erreicht werden, dass sie explizit in die Anwendung installiert werden oder dadurch, dass sie in den Ab-

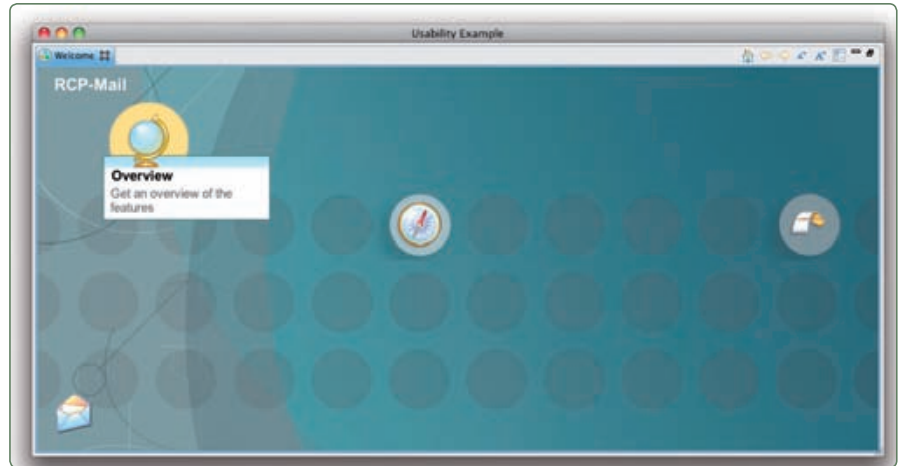


Abb. 1: Welcome Page

hängigkeiten der Anwendung aufgelistet sind und daher beim Bauen aus Eclipse heraus mit installiert werden. In unseren Beispielen haben wir aus Gründen der leichteren Nachvollziehbarkeit jeweils alle benötigten Plug-ins deklariert.

Um Welcome Pages wie in der Eclipse-IDE zu erhalten, werden zwei Plug-ins benötigt. Dabei stellt *org.eclipse.ui.intro* die benötigten Extension Points bereit, *org.eclipse.ui.intro.universal* die Standardimplementierung, wie sie auch in der IDE Anwendung findet.

Die Welcome Pages werden mit ein paar Deklarationen in der *plugin.xml*-Datei eingebunden. Dazu muss der Extension Point *org.eclipse.ui.intro* benutzt werden:

```
<extension point="org.eclipse.ui.intro">
  <introProductBinding
    introId="org.eclipse.ui.intro.universal"
    productId="com.bredexsw.examples.mailbase.
                                     product">
  </introProductBinding>
</extension>
```

Anzeige

Anzeige

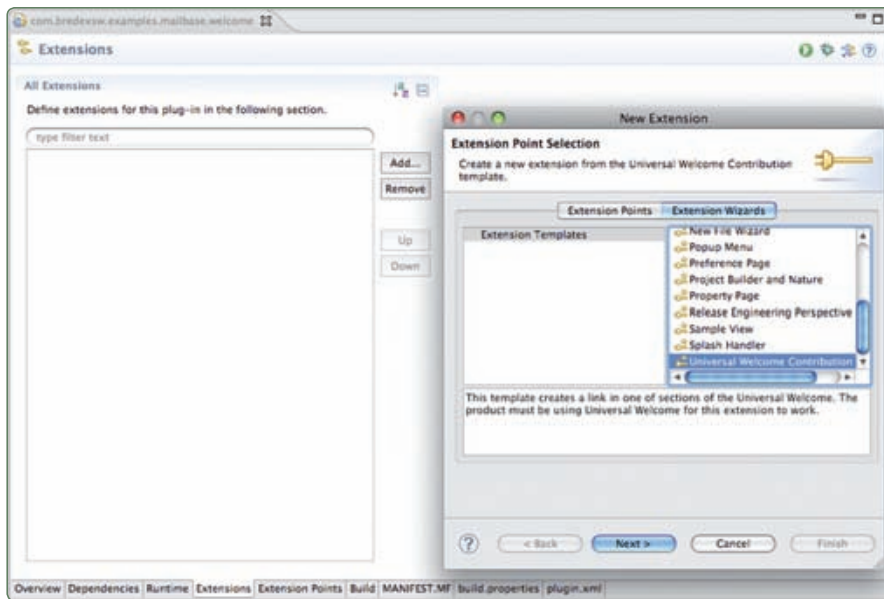


Abb. 2: Erzeugen einer Welcome Contribution

Ergebnis ist aber sicherlich nicht beeindruckend. Daher werden wir jetzt einige optische Verbesserungen vornehmen, bevor wir uns den Inhalten zuwenden. Dazu müssen einige Properties im *product*-Element von *org.eclipse.core.runtime.products* gesetzt werden (Tabelle 2 und Listing 2).

Durch diese Properties wird das Aussehen der Welcome Pages vorgegeben, jedoch noch kein möglicher Inhalt. Der Inhalt wird leider nicht im *plugin.xml* festgelegt, sondern in der *plugin_customization.ini* (im Plug-in-Root-Verzeichnis):

```
# parameters for the welcome screen
org.eclipse.ui.intro.universal/INTRO_ROOT_
    PAGES=overview,firststeps
org.eclipse.ui.intro/INTRO_THEME=org.eclipse.ui.intro.
    universal.circles
```

Im Element *introProductBinding* wird die Zuordnung von Welcome Page und Produkt hergestellt. Die im Attribut *introId* angegebene Kennung *org.eclipse.ui.intro.universal* verweist auf die im gleichnamigen Plug-in realisierte Standardimplementierung der Welcome Pages, die wir durchgehend verwenden werden. Falls es notwendig sein sollte, eine eigene Implementierung der Welcome Pages zu erstellen, so muss diese Kennung entsprechend auf die eigene Implementierung verweisen. Wir raten aber dazu, immer die Standardimplementierung zu benutzen, um auf ein bekanntes und bewährtes Modell zu setzen.

Das Attribut *productId* gibt an, dass diese Welcome Pages an ein bestimmtes Produkt gebunden sind. Der Wert hierfür ist etwas heikel: Er muss genau zu ande-

ren Kennungen passen, da sonst die Zuordnung nicht erfolgen kann. Dies würde in die hässliche Lage führen, dass die Anwendung völlig normal und ohne Fehlermeldungen startet, die mühsam erstellten Welcome Pages jedoch völlig ignoriert würden. Daher sollten die folgenden Werte immer passend gesetzt werden (Tabelle 1). Zusammengefasst in der *plugin.xml* sieht das dann wie in Listing 1 gezeigt aus. Diese Einstellungen sind technisch nicht zwingend nötig. Sie werden hier aber so empfohlen, da sie das Zusammenspiel aller Komponenten erleichtern.

Erster Eindruck

Prinzipiell ist damit alles getan, um eine rudimentäre Welcome Page anzuzeigen. Probieren Sie es einfach mit der Launch Configuration *base_welcome* aus. Das

Die *INTRO_ROOT_PAGES* listen dabei alle gewünschten Einstiegspunkte auf. Komplet existieren *overview*, *firststeps*, *tutorials*, *samples*, *whatsnew*, *migrate* und *webresources*. Das *INTRO_THEME* gibt die Darstellung vor, ohne die Inhalte zu beeinflussen. Vergessen Sie nicht, die neu angelegte *plugin_customization.ini* im *MANIFEST.MF* als Teil der Anwendung zu spezifizieren. Beide Werte lassen sich auch durch den Anwender im Nachhinein über den Konfigurationsdialog (*Customize Page*) der Welcome Pages einstellen. Damit ist die Welcome Page (gestartet über *base_welcome_branded*) schon recht präsentabel (Abb. 1).

Auf den Inhalt kommt es an

Der Inhalt, der hinter den einzelnen Punkten (im Beispiel *Overview* und *First*

Attribut	Wert	Beispiel
Plug-in-ID der Anwendung (PID)	Eindeutige Kennung entsprechend Package-Konventionen	<i>com.bredexsw.examples.mailbase</i>
Extension Point <i>org.eclipse.core.runtime.applications</i> , Attribut <i>id</i>	<PID>.application	<i>com.bredexsw.examples.mailbase.application</i>
Extension Point <i>com.bredexsw.examples.mailbase.product</i> , Attribut <i>id</i>	<PID>.product	<i>com.bredexsw.examples.mailbase.product</i>
Extension Point <i>com.bredexsw.examples.mailbase.product</i> , Element <i>product</i> , Attribut <i>application</i>	<PID>.application	<i>com.bredexsw.examples.mailbase.application</i>

Tabelle 1: Attribute für Welcome Pages

Property	Bedeutung	Beispiel
<i>introTitle</i>	Titel der Welcome Page	RCP Mail
<i>introBrandingImage</i>	Symbol für die Anwendung	<i>product:icons/mail.ico</i>
<i>introBrandingImageText</i>	Ersatzweise Text anstelle des Symbols	RCP Mail

Tabelle 2: Branding der Anwendung

Steps) steht, sollte durch ein neues Plug-in hinzugefügt werden. Dazu wird ein leeres Resource-Plug-in angelegt. In diesem Plug-in werden nur die für den Inhalt nötigen Konfigurations- und Datendateien hinterlegt, Programmierung ist nicht nötig. Dies erleichtert es natürlich, die Umsetzung durch Teammitglieder aus dem Bereich Dokumentation und Support durchführen zu lassen. Beim Anlegen des Plug-in-Projekts muss darauf geachtet werden, dass keine Optionen ausgewählt sind, die für eine Programmierung oder für das Lifecycle-Management (z.B. *Activator*-Klasse) relevant sind. Die Plug-in-ID wird entsprechend der oben genannten Konvention gewählt, für das Beispiel also *com.bredexsw.examples.mailbase.welcome*.

Welcome Pages werden als Extension Point durch einen Wizard unterstützt. Dazu muss in der Konfiguration

durch Editieren der *MANIFEST.MF* das Plug-in als Singleton definiert werden (im Editor im Tab *Overview*). Danach gehen wir den leichten Weg und benutzen den vorhandenen Wizard, um die restliche Konfiguration zu erstellen. Dazu muss beim Zufügen einer Extension durch den *Extensions Wizard* eine *Universal Welcome Contribution* ausgewählt werden (Abb. 2).

Der Wizard legt eine rudimentäre Infrastruktur für die Inhalte an. Es werden die Konfigurationsdateien für die Verknüpfungen und für Cascading Style Sheets erzeugt. Für die im Wizard voreingestellten Werte (Directory *intro*, Target Page *Overview*) wird folgende Struktur erstellt:

- <plug-in root>
- *intro*
- *sample.xml*
- *css*
- *sample.css*
- *graphics*
- ... icons etc.

Listing 1

```
<extension
  id="com.bredexsw.examples.mailbase.application"
  point="org.eclipse.core.runtime.applications">
<application>
  <run
    class="com.bredexsw.examples.base.
                                     Application">
  </run>
</application>
</extension>
<extension
  id="com.bredexsw.examples.mailbase.product"
  point="org.eclipse.core.runtime.products">
<product
  application="com.bredexsw.examples.mailbase.
                                     application"
  name="Usability Example">
<property
  name="aboutText"
  value="RCP Mail template created by PDE">
</property>
<property
  name="windowImages"
  value="icons/sample2.gif">
</property>
<property
  name="aboutImage"
  value="product_lg.gif">
</property>
</product>
</extension>
<extension
  point="org.eclipse.ui.intro">
<introProductBinding
  introId="org.eclipse.ui.intro.universal"
  productId="com.bredexsw.examples.mailbase.
                                     product">
</introProductBinding>
</extension>
```

Listing 2

```
<extension
  id="com.bredexsw.examples.mailbase.product"
  point="org.eclipse.core.runtime.products">
<product
  application="com.bredexsw.examples.mailbase.
                                     application"
  name="Usability Example">
<property
  name="aboutText"
  value="RCP Mail template created by PDE">
</property>
<property
  name="windowImages"
  value="icons/sample2.gif">
</property>
<property
  name="aboutImage"
  value="product_lg.gif">
</property>
<property
  name="introTitle"
  value="RCP-Mail">
</property>
<property
  name="introBrandingImage"
  value="product:icons/mail.ico">
</property>
<property
  name="introBrandingImageText"
  value="RCP Mail">
</property>
</product>
</extension>
```

Anzeige

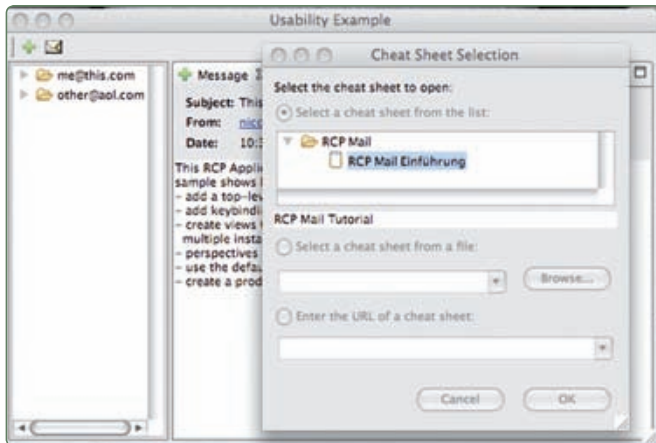


Abb. 3: Cheat-Sheet-Auswahldialog

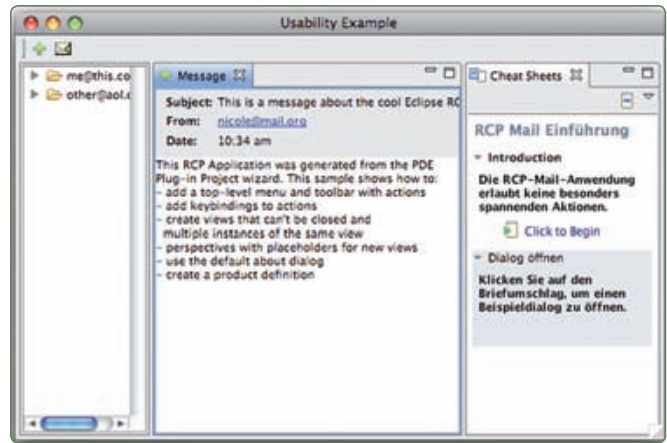


Abb. 4: RCP-Mail-Cheat-Sheet

Leider wird vom Wizard der Export des Plug-ins nicht mit angepasst, daher muss noch das *intro*-Verzeichnis zu den benötigten Artefakten hinzugefügt werden (im Tab *Build*). Wenn dieses Plug-in beim Start der Anwendung mit angegeben wird, ist jetzt ein Beispieltext für die Sparte *Overview* vorhanden (die Launch Configuration *base_welcome_content* zeigt dies an). Die Datei *sample.xml* enthält die Informationen, die auf den Inhalt der Welcome Pages verweisen (Listing 3).

Wichtig sind die Verweise auf die Cascading Style Sheets im Attribut *style* und die Einordnung in den gesamten Kontext über das Attribut *path*. Die CSS benutzen die generierte *sample.css*. Der Wert *overview/@* gliedert diese Contribution in die Kategorie *Overview* ein. Sobald man eine Navigation über mehrere Unterseiten vorsieht, kann dieser Parameter als Verweis

in der Navigation benutzt werden. Die in *content-group* hinterlegten Informationen sind exemplarisch für eine mehr oder weniger aufwändige HTML-Darstellung zu sehen. Die simple Anzeige eines kurzen Textes und ein Verweis auf eine HTML-Seite werden über das *link*-Target erreicht. Die Erstellung von ansprechenden HTML-Daten überlassen wir Ihrer Dokumentationsgruppe als Übung.

Was geht noch?

Aus Gründen der Übersichtlichkeit haben wir auf die Internationalisierung der Texte verzichtet. Die in RCP etablierten Mechanismen über Properties-Dateien und die %-Notation funktionieren für Parameter in der *plugin.xml*. Für die Inhalte sind entsprechend Verzeichnisse mit den Kennungen für Sprache, Land und Kulturkreis möglich. Details hierzu finden Sie in unserem Walk-Through [2].

Komplett ignoriert haben wir die Möglichkeit, Welcome Pages auch in SWT zu erstellen. Dies ist aus unserer Sicht auch nicht sinnvoll, da Dokumentation und Programmierung sich normalerweise in disjunkten Universen abspielen.

Cheat Sheets: nicht mogeln, sondern helfen

Als allerersten Schritt möchten Benutzer normalerweise eine Anwendung ausprobieren, und zwar ohne erst einmal lange Anweisungen lesen zu müssen. Die Cheat Sheets (Spickzettel) sind interaktive Tutorials, die man direkt im Programm aufruft und durcharbeitet. Sie sind eine sehr geeignete Stelle, um die wichtigsten Abläufe der Anwendung und die Best Practices zu erläutern. Die Cheat Sheets bestehen aus Aufgaben, Schritten und Sub-Schritten, die der Benutzer selbst nachvollziehen oder vom Programm

Listing 3 ✕

```
<?xml version="1.0" encoding="utf-8"?>
<introContent>
<extensionContent id="combredexwexamplesmail
    basewelcme-introExtension"
    style="css/sample.css"
    name="Sample Extension"
    path="overview/@">
<group style-id="content-group"
    id="combredexwexamplesmailbasewelcme-intro
        Link-group">
<link label="Mail-Überblick" url="http://www.
    bredex.de/"
    id="combredexwexamplesmailbasewelcme-intro
        Link"
    style-id="content-link">
<text>Die RCP-Mail-Anwendung ist ...
</text>
</link>
</group>
</extensionContent>
</introContent>
```

Listing 4 ✕

```
<?xml version="1.0" encoding="UTF-8"?>
<cheatsheet
    title="RCP Mail Einführung">
<intro>
<description>
<b>Die RCP-Mail-Anwendung erlaubt keine
    besonders spannenden Aktionen.</b>
</description>
</intro>
<item
    title="Dialog öffnen">
<description>
<b>Klicken Sie auf den Briefumschlag,
    um einen Beispieldialog zu öffnen.</b>
</description>
</item>
</cheatsheet>
```

Listing 5 ✕

```
<extension
    point="org.eclipse.ui.cheatsheets.
        cheatSheetContent">
<category
    name="RCP Mail"
    id="base_cheatsheet.category1151466294">
</category>
<cheatsheet
    name="RCP Mail Einführung"
    category="base_cheatsheet.category
        1151466294"
    contentFile="$nls/MailRCPIntro.xml"
    composite="false"
    id="base_cheatsheet.cheatsheet327409891">
<description>
    RCP Mail Tutorial
</description>
</cheatsheet>
</extension>
```



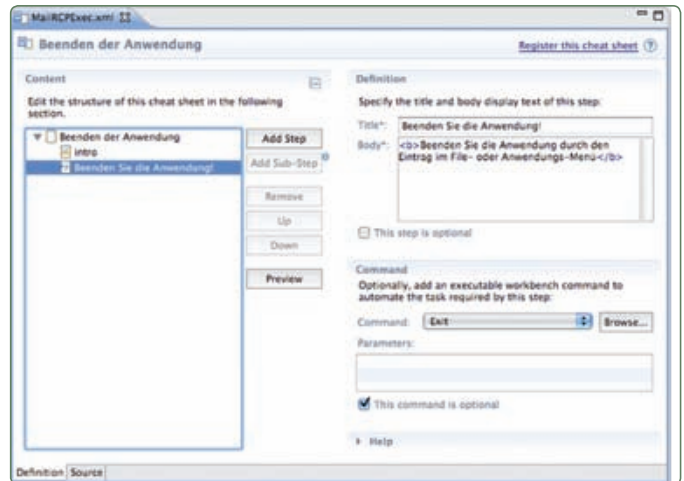
ausführen lassen kann. Schritte können voneinander abhängig oder unabhängig sein, können als optional gekennzeichnet werden und sogar das Hilfesystem verlinken. Kleinere Cheat Sheets lassen sich zu „Composite Cheat Sheets“ zusammenführen, um größere Aufgaben abzudecken. Wichtig bei den Cheat Sheets ist es, immer den einfachsten und eindeutigsten Weg zu zeigen und sich auf die Hauptfunktionalitäten zu beschränken. Auch für Anwendungsentwickler sind Cheat Sheets hilfreich: Wenn bei der Erstellung eines Cheat Sheets auffällt, dass eine Aufgabe sehr schwierig durchzuführen ist, deutet dies auf den Bedarf für eine grundsätzlichere Überarbeitung des Bedienkonzepts hin. Wie auch die Welcome Pages sind Cheat Sheets ohne jede Programmierung zu erstellen. Sie lassen sich dynamisch durch ein Plug-in jeder RCP-Anwendung hinzufügen. Dazu ist wiederum ein leeres Plug-in-Projekt nötig. Für unser Beispiel erstellen wir ein Plug-in mit der ID *com.bredexsw.examples.mailbase.cheatsheets*. Das Plug-in benötigt die Plug-ins *org.eclipse.ui*, *org.eclipse.ui.cheatsheets* und *org.eclipse.help.ui* für die weiteren Schritte.

Wie spickt man?

Dass Cheat Sheets ein wichtiges Mittel zur Benutzerhilfe sind, kann man daran sehen, dass es für sie eine gute Unterstützung in der IDE gibt. Ein Cheat Sheet kann durch einen Wizard angelegt werden, der viele Details gleich mit erledigt. Nach der Ausführung des *New-Wizards* für Cheat Sheets (in der Kategorie *User Assistance*) ist ein Rahmen für ein Cheat Sheet erzeugt und im zugehörigen Editor geöffnet. Für das Beispiel wird zuerst ein *Simple Cheat Sheet* in der Datei *MailRCPIntro.xml* erzeugt. Die Bearbeitung dieser Datei sollte ausschließlich über diesen Editor erfolgen, da dann alle IDs und Referenzen automatisch gepflegt werden. Cheat Sheets bestehen aus wenigen Komponenten: einer Überschrift, einer Einführung und den Schritten des Tutorials. Die *MailRCPIntro.xml* sieht wie in Listing 4 aus:

Im Cheat-Editor kann man ein Cheat Sheet im Framework registrieren. Dazu ist nur ein Klick auf den entsprechenden Link nötig. Im dann folgenden Dialog kann und sollte man eine Kategorie für die Anwendung anlegen. Die dadurch erzeugte Strukturierung ist speziell dann hilfreich, wenn Teile der Anwendung als

Abb. 5: Cheat-Sheet-Editor mit Command



Plug-in an anderer Stelle benutzt oder andere Plug-ins mit Cheat Sheets in die eigene Anwendung integriert werden sollen. Die Registrierung trägt die XML-Datei in den Extension Point *cheatSheetContent* ein. Sollte dies (übrigens ohne jede Fehlermeldung) nicht passieren, wurden die oben genannten Abhängigkeiten (Plug-ins) nicht eingetragen (Listing 5).

Die IDs werden vom Editor automatisch generiert und verwaltet, es gibt also keinen Grund, sie speziell anzupassen. Der Verweis *contentFile* enthält schon die Vorbereitung auf Internationalisie-

rung, *\$nl\$* wird gegebenenfalls durch ein entsprechendes Verzeichnis ersetzt.

Mit den bisherigen Schritten ist ein Cheat Sheet komplett erstellt. Wenn das Plug-in nun der Anwendung hinzugefügt wird, sollte man eigentlich auf das Cheat Sheet zugreifen können. Das funktioniert allerdings nur, wenn die Anwendung schon eine passende Aktion bereitstellt. In der IDE ist dies ein Unterpunkt des Help-Menüs. Für die Beispielanwendung muss jedoch noch eine entsprechende Bedienung eingebaut werden. Auch dies funktioniert rein deklarativ.

Anzeige

Anzeige

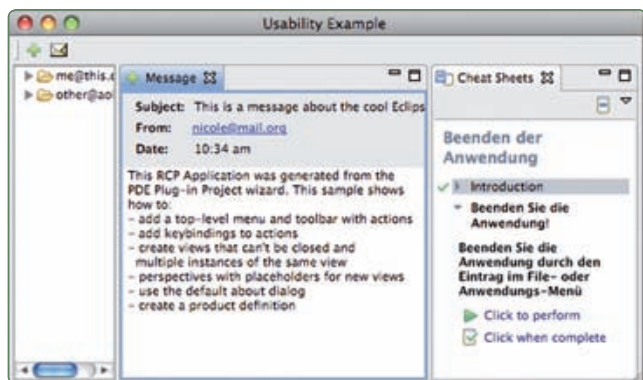


Abb. 6: Cheat Sheet mit automatischer Ausführung

das Beenden der Anwendung (Starten mittels *base_cheatsheet_commands*).

Die Anwendung zeigt dann ein Cheat Sheet, dessen Schritte durch Selektion automatisch ausgeführt werden können. Auch ein Mischen zwischen manueller und automatischer Ausführung ist sowohl von der Definition als auch bei der Ausführung möglich (Abb. 6).

Resümee

Wir schreiben Software für Benutzer. Mit dieser Software sollen Aufgaben einfacher erledigt werden können und nicht das Leben der Benutzer erschwert werden. Da die Benutzer eventuell unerfahren sind, muss die Software verständlich und benutzerfreundlich sein. Die Welcome Pages und Cheat Sheets sind zwei einfache Möglichkeiten, die Benutzerfreundlichkeit der Anwendung zu erhöhen. Weitere Hilfestellungen wie kontextsensitive Hilfe und die Eclipse Control Decoration werden in einem späteren Artikel behandelt.

Wie die Software selbst, muss die Benutzerfreundlichkeit immer mit den Kundenanforderungen wachsen. Wo gibt es die meisten Support-Anfragen? Was wird zu häufig missverstanden oder nicht gefunden? Reicht ein neues Cheat Sheet aus, oder muss an der Software selbst gearbeitet werden? Oft genügen ein paar Gedanken zum richtigen Zeitpunkt, damit ein neues Feature in der Anwendung gut assimiliert wird und sich intuitiv bedienen lässt.



Alexandra Imrie hat einen Abschluss als Master of Phonetik/Phonologie. Sie beschäftigt sich bei der BREDEX GmbH mit den Konzepten und der Planung für das Testwerkzeug GUIDancer sowie mit Dokumentation, Kundensupport und Demos.



Achim Lörke ist in der Geschäftsführung der BREDEX GmbH für die Auswahl von technischen Richtlinien und Architekturen zuständig. Er betreut die Mitgliedschaft in der Eclipse Foundation und die Open-Source-Aktivitäten der Firma. Zusätzlich leitet er die GUIDancer-Entwicklung und ist (soweit es die Zeit zulässt) Eclipse-Committer.

>> Links & Literatur

- [1] Beispielanwendung: <http://www.bredex.de/de/downloads/whitepapers.html>
- [2] Detaillierter Walk-Through zur Benutzerfreundlichkeit (englisch) <http://www.bredex.de/de/downloads/whitepapers.html>

Dazu muss ein ActionSet als Extension Point deklariert werden.

```
<extension point="org.eclipse.ui.actionSets">
<actionSet
  label="Cheat Sheets"
  visible="true"
  id="org.eclipse.ui.cheatsheets.actionSet">
<action
  label="Cheat Sheets..."
  class="org.eclipse.ui.cheatsheets.CheatSheetExtension
    Factory:helpMenuAction"
  menubarPath="help/group.tutorials"
  id="org.eclipse.ui.cheatsheets.actions.
    CheatSheetHelpMenuAction" />
</actionSet>
</extension>
```

Die Werte für die IDs, Classes usw. sind extrem kritisch, ein kleiner Tippfehler

führt u. U. zu vielen Stunden der Suche nach dem Grund, warum kein Menüeintrag erscheint. Zur Erhaltung der geistigen Gesundheit sollte dieses Segment aus dem Beispiel kopiert und direkt in die *plugin.xml* eingefügt werden. Nun sind der Menüeintrag und die Cheat Sheets verfügbar. Mittels der Launch Configuration *base_cheatsheet* ergibt sich die Situation wie in den Abbildungen 3 und 4 gezeigt.

Um den Komfort noch zu steigern, können *Commands* der Anwendung auch direkt aus dem Cheat Sheet ausgelöst werden. Dazu müssen die *Command*-Klasse und ihre Parameter angegeben werden. Auch dies lässt sich komfortabel im Editor erledigen (Abb. 5). Im Beispiel beschränken wir uns auf

Aufbau der Beispielanwendung

Die Eclipse-Projekte für die verschiedenen Stufen der Beispielanwendung können als Zip-Archiv heruntergeladen [1] und in einen Workspace installiert werden (per FILE | IMPORT...).

Projekte

Die eigentliche Anwendung steht jeweils in den Projekten, die mit *base* beginnen. Sie unterscheiden sich nur durch geänderte Konfigurationen (*plugin.xml*, *MANIFEST.MF*). Zur besseren Unterscheidung in den Launch Configurations und beim Erstellen einer Anwendung ist jeweils die Major-Version um eins erhöht.

- base
- base_welcome
- base_welcome_branded
- In *welcome_pages* sind die Konfigurationen und Inhalte der Beispielseiten hinterlegt
- *welcome_pages*
- Die Projekte *cheatsheet...* enthalten die Konfigurationen und Inhalte der Cheat Sheets
- *cheatsheet*
- *cheatsheet_commands*

- Inhalt von *configurations* sind Launch Configurations (s. u.)
- *configurations*

Launch Configurations

Zum leichten Ausprobieren der verschiedenen Ausbaustufen sind Launch Configurations beigefügt. Sie stehen im Projekt *configurations* im Verzeichnis *launch-configs*. Abweichend von der Benennung im Text ist der Name der Launch Configuration jeweils aus der unterstützten Plattform und dem „eigentlichen“ Namen zusammengesetzt. So heißt *base* beispielsweise tatsächlich u. a. *galileo_osx_cocoa_base*. Probieren Sie einfach aus, was für Ihre Plattform vorhanden ist, und passen Sie ggf. die Konfigurationen an.

Walk-Through

Ein Walk-Through (d. h. Schritt für Schritt und reich bebildert) zum Einbau von Bedienhilfen ist (auf Englisch) unter [2] verfügbar. Dort wird eine RCP-Anwendung „aus dem richtigen Leben“ um entsprechende Konstrukte erweitert.