trivadis

Client-Entwicklung mit JFC oder SWT?

AUTOR

Andreas Spall
Orientation in Objects GmbH

Veröffentlicht am: 18.11.2003

ABSTRACT

Ansprechende Java-Clients konnten bisher nur mit Swing entwickelt werden. Das Eclipse-Projekt hat mit SWT die Entwicklung von graphischen Benutzeroberflächen um eine weitere Möglichkeit bereichert. JFC oder SWT heißt jetzt die Frage?

Dieser Artikel erleichtert dem Leser die Entscheidung, indem er die Gemeinsamkeiten und Unterschiede der beiden Technologien betrachtet.

Bei der Entwicklung der Eclipse Plattform entschieden sich das Team von OTI (Object Technology International) und IBM gegen die JFC. Statt dessen entschlossen sie sich zum Erstellen der GUI eine eigene API zu entwickeln und diese, ebenso wie die Plattform auch, als Open Source frei zu geben. Diese neue API trägt den Namen Standard Widget Toolkit (SWT) und stellt GUI-Komponenten (Widgets), wie zum Beispiel Buttons, Bäume oder Tabellen über native Komponenten des Betriebssystems zur Verfügung.

) Schulung)

) Beratung)

) Entwicklung)

) Artikel)

Trivadis Germany GmbH

Weinheimer Str. 68
D-68309 Mannheim

Tel. +49 (0) 6 21 - 7 18 39 - 0
Fax +49 (0) 6 21 - 7 18 39 - 50

www.oio.dekontakt@trivadis.com

DIE JAVA BOARDMITTEL

Im Folgenden werden die beiden Hauptbestandteile der JFC, AWT und Swing, betrachtet. Weitere Bestandteile wie zum Beispiel die Accessibility API oder die 2D Graphics and Imaging API sind für diesen Vergleich zwischen AWT / Swing und SWT nicht von Interesse und werden aus diesem Grund nicht betrachtet.

AWT

Das Abstract Window Toolkit (AWT), verwendet nur Widgets, die auf allen Plattformen unterstützt werden. Dies bedeutet, dass Komponenten wie Bäume oder Tabellen im AWT nicht vohanden sind. Über einen LayoutManager ist es möglich, einzelne GUI-Komponenten relativ zu platzieren. Da diese Komponenten aber von Betriebssystem zu Betriebsystem ein unterschiedliches Aussehen besitzen, kann die Oberfläche bei einem Betriebsystem optimal aussehen, bei einem anderen aber inakzeptabel sein. AWT Komponenten unterliegen wie andere Objekte der Garbage Collection. Dadurch werden die vom Programm nicht mehr benötigten Ressourcen von der JVM freigegeben, ohne dass der Programmierer dies steuern muss. Im Grunde ist das AWT eine einheitliche objektorientierte Schicht über das jeweilige native Fenstersystem der Plattform und stellt diese über eine Peer Referenz innerhalb der Plattform dar.

SWING

AWT war, mit 6 Wochen Entwicklungszeit bis zur ersten Release, eine Schnellgeburt und hatte dementsprechend einige Schwächen. Diese Schwächen wurden durch die Internet Foundation Classes (IFC) von Netscape zunächst bereinigt. Um eine Spaltung der Javagemeinde zu vermeiden, schloss sich Sun mit Netscape und anderen Unternehmungen zu einem Implementierungskonsortium zusammen. Das Framework Swing bzw. die JFC sollte dem GUI-Entwickler das Beste von AWT und IFC in einer umfassenden Werkzeugsammlung zur Verfügung stellen. In Swing wurden die graphischen Komponenten in Heavyweight- und Lightweight-Komponenten aufgeteilt. Unter Heavyweight-Komponenten versteht man Anwendungsfenster und Dialogfenster, die wie in AWT einen Peer als Implementationshilfe besitzen. Lightweight-Komponenten hingegen (JButton, JTextField usw.) wurden vollständig in Java implementiert und müssen von Heavyweight-Komponenten zu ihrer Darstellung aufgenommen werden.

WAS GENAU SIND SWT UND JFACE

Im Gegensatz zu AWT verwendet SWT nicht nur Widgets, die auf allen Zielplattformen zur Verfügung stehen, sondern es werden auch plattformspezifische Widgets angeboten. In der Windows-SWT Welt ist dies beispielsweise der Zugriff auf OLE (Object Linking and Embedding) Dokumente über das `org.eclipse.swt.ole.win32.OleFrame`. Dadurch ist es SWT-Entwicklern möglich, über OLE zum Beispiel den Internet Explorer anzusprechen.

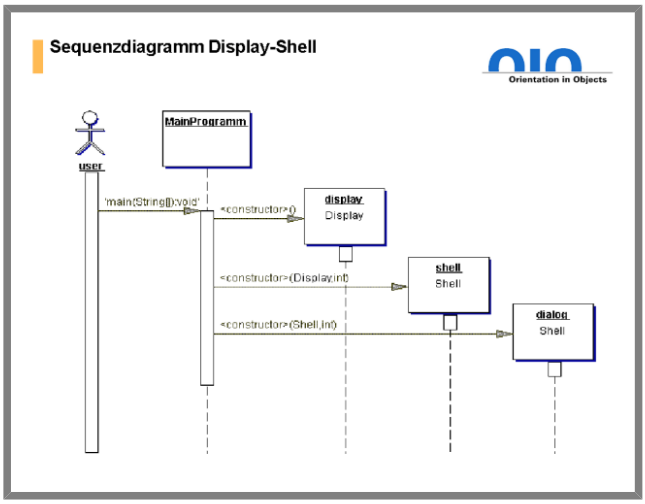

Abbildung 1: Sequenzdiagramm Display-Shell

Wie oben beschrieben sind Widgets graphische Interaktionsobjekte. Das Displayobjekt dient der Verbindung von SWT mit dem zu Grunde liegenden GUI des Betriebssystems. Über dieses Display ist es möglich, Shells als Fenster darzustellen. Ist eine Shell als Kind einer anderen Shell realisiert, wird sie dem User als Dialog-Fenster nur dann dargestellt, wenn die Parent-Shell existiert. Dazu wird die Parent-Shell dem Kind in dessen Konstruktor bekanntgegeben.

Beim Erzeugen eines Widgets, muss man diesem seinen Style bekannt geben. Der Style kann nachträglich nicht mehr verändert werden, dazu später mehr.

FREIGABE VON RESSOURCEN

Die Betriebssysteme unterhalb von SWT benötigten einen expliziten Belegungshinweis und eine explizite Freigabe von Ressourcen. Die Freigabe wird in SWT durch die Methode `dispose()` vorgenommen. Aus diesem Grund muss man immer der Regel folgen:

You create the Object, you must dispose it.

Für diese Regel gibt es, wie für viele andere auch, Ausnahmen:

- Wenn man ein Objekt erhält, ohne dessen Konstruktor aufzurufen, muss man es nicht freigeben.
- Wenn man für eine Shell `dispose()` aufruft, werden alle Widget-Kinder rekursiv freigegeben. In diesem Fall müssen die Kindobjekte nicht manuell freigegeben werden.

JFACE

JFace erweitert die Funktionen von SWT und dient als Schnittstelle zwischen SWT und der Eclipseplattform. Über JFace ist eine Trennung von Modell und Darstellung (ähnlich dem MVC bei Swing) möglich. Da JFace nicht als eigenständige Distribution erhältlich ist, wird im Folgenden nicht darauf eingegangen.

BEISPIEL

Um die unterschiedlichen Arbeitsweisen von Swing und SWT zu verdeutlichen, wurde ein einfaches Programm in beiden Technologien implementiert (Quellcode siehe [1]). Aufgabe dieses Programmes ist es, Texteingaben über ein Textfeld zu ermöglichen und diesen Text nach dem Drücken eines "Add"-Buttons einer Liste hinzuzufügen. Wird ein Text in der Liste selektiert, soll dieser Text im Textfeld erscheinen.

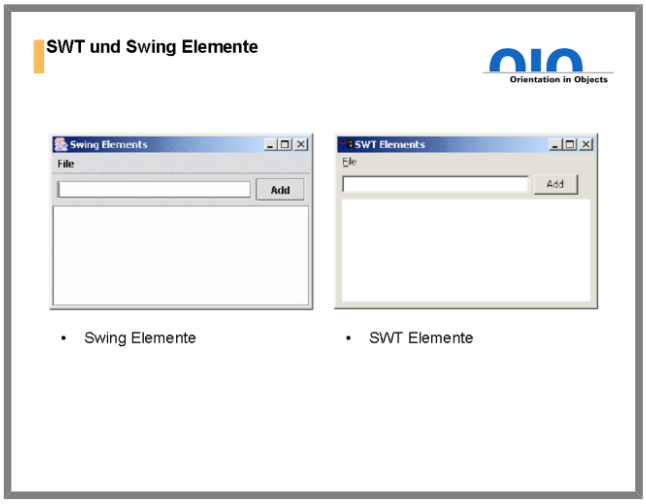

Abbildung 2: Swing und SWT Elemente

PROGRAMM MIT SWING

Am Anfang wird ein JFrame-Objekt innerhalb der `main()` -Methode erzeugt. Dieses JFrame dient als Hauptfenster der Applikation (siehe Abbildung 3; Punkt 1). Dem Hauptfenster wird über die Methode `setContentPane()` ein JPanel-Objekt übergeben, über dieses contentPane ist es möglich, dem Fenster entsprechend dem festgelegten Layout Komponenten wie Buttons, Textfelder und Listen anzufügen.

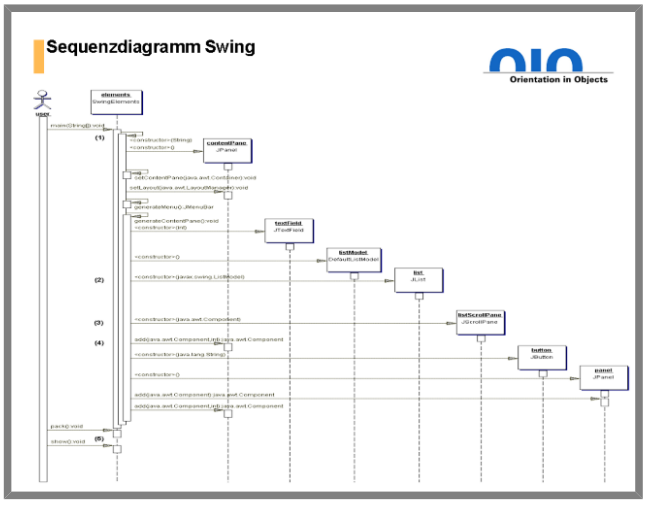

Abbildung 3: Sequenzdiagramm Swing Beispiel

Wie man dem Sequenzdiagramm entnehmen kann, wird dem Konstruktor von JList (siehe Abbildung 3; Punkt 2) beim Erzeugen ein DefaultListModel übergeben. Über dieses DefaultListModel können nun in der Liste Einträge getätigt werden. Das JList-Objekt ist nicht für die Einträgeverwaltung sondern nur für die Darstellung zuständig.

Damit in der Liste gescrollt werden kann, müssen wir diese in eine JScrollPane legen (siehe Abbildung 3; Punkt 3).

Über die Methode `add(Component)` übergeben wir die listScrollPane zusammen mit Angaben für das Layoutmanagement der contentPane (siehe Abbildung 3; Punkt 4). Der Button und das Textfeld werden in einem JPanel (panel) zusammengefaßt und über dieses JPanel an die contentPane übermittelt.

Nachdem die Darstellung festgelegt wurde, muss man der Applikation noch Leben einhauchen. Dies geschieht über sogenannte EventListener (`java.util.EventListener`). Diese haben die Aufgabe, auf Aktionen des Benutzers so zu reagieren, wie dies der Entwickler der Anwendung wünscht. Über die abstrakte Klasse AbstractAction ist es möglich, eine Aktion zu kapseln und diese an unterschiedliche Objekte zu übergeben. Steuermechanismen können so zentralisiert werden.

Die Applikation wird dem Benutzer über die Methode `frame.show()` dargestellt (siehe Abbildung 3; Punkt 5). Das Beenden der Applikation wird über einen `WindowListener` ermöglicht, dieser ruft an der Klasse System die Methode `exit()` auf und dies führt zum Beenden der VM.

PROGRAMM MIT SWT

Am Anfang wird ein Display-Objekt innerhalb des Konstruktors der Applikation erzeugt. Dieses Display dient als Verbindung der SWT-Anwendung mit dem zu Grunde liegenden GUI des Betriebssystems (siehe Abbildung 4; Punkt 1). Als nächstes wird ein Shell-Objekt erzeugt, dabei wird das Display-Objekt dem Konstruktor übergeben. Dies hat zur Folge, dass diese Shell zu einem Hauptfenster der Applikation wird. Nun ist es möglich, dem Fenster, entsprechend dem festgelegten Layout, den Button, das Textfeld und die Liste anzufügen.

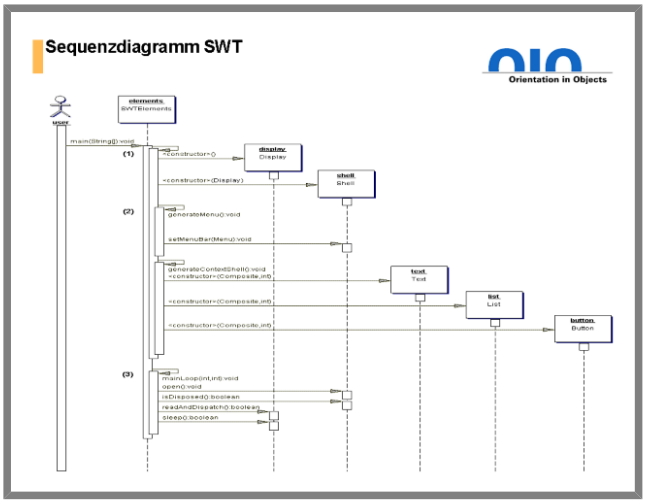

Abbildung 4: Sequenzdiagramm SWT Beispiel

Wie man dem Sequenzdiagramm entnehmen kann, wird dem Text (siehe Abbildung 4; Punkt 2) beim Erzeugen ein Composite und ein int-Wert übergeben. Dieses Composite stellt den Parent des SWT-Objektes dar. Über den int-Wert ist es möglich, den Style des zu erzeugenden Objektes einmalig anzugeben. Die Größe kann man über die Methode `setLayoutData()` festlegen. Die Klassen Button und List bieten auch einen Konstruktor, über welchen man einen Parent und das Layout übergeben kann, an. Über die folgende Zeile kann im Konstruktor der Liste angegeben werden, dass die Liste sowohl vertikal als auch horizontal scrollfähig ist:

```
new List(shell, SWT.V_SCROLL | SWT.H_SCROLL);
```

Beispiel 1: Listenkontruktor

Ein `add()`, wie bei Swing, an der Shell aufzurufen und dieser dadurch die Komponenten bekannt zu geben, ist durch das Übergeben der Shell im Konstruktor nicht mehr nötig. Die Komponenten sind bereits in einem Parent-Child-Verhältnis miteinander verbunden.

Nachdem die Darstellung nun erreicht wurde, muss man der Applikation noch Leben einhauchen. Dies Geschieht wie in AWT/Swing über sogenannte EventListener (`java.util.EventListener`) bzw. dessen SWT-Erbling SWTEventListener (`org.eclipse.swt.internal.SWTEventListener`).

Nun kann man die Applikation dem Benutzer über den Aufruf `shell.open( )` zur Darstellung bringen (siehe Abbildung 4; Punkt 3). Wird die Applikation vom User beendet, kann dies die Applikation über die Methode `isDisposed( )` der Shell erfragen. Möchte man die Applikation programmatisch beenden, kann man an der Shell die Methode close() aufrufen. Dadurch werden an allen Child-Objekten dispose() aufgerufen und die verwendeten Systemressourcen wieder freigegeben.

Wenn man eine Applikation unter Zuhilfenahme des SWT-Frameworks erstellen möchte, muss man natürlich sicherstellen, dass der Benutzer das benötigte swt.jar und dessen Systembibliotheken besitzt. Eine Java WebStart Anwendung, wie sie in [2] realisiert wurde, könnte dies automatisch bewerkstelligen.

FAZIT

Unter Betrachtung der Beispiele [1] kann man sagen, dass die Swing Variante Aufgrund der Zeilen kleiner als das SWT-Beispiel ist, aber mehr Klassen verwendet. Von der Lesbarkeit des Quellcodes nehmen und geben sich die beiden Kontrahenten im direkten Vergleich nichts. In Summe kann man festhalten.

Möchte man einen Client schreiben,

- dessen Look and Feel man selbst bestimmen kann, sollte man weiterhin bei Swing bleiben.
- bei dem auch GUI - Objekte vom GC verwaltet werden, sollte man weiterhin bei Swing bleiben.
- dessen Look and Feel zu 100% den Anwendungen entspricht, die speziell für die Plattform geschrieben wurden, sollte man SWT oder die Programmiersprache der Plattform wählen.
- der sich bei Benutzerinteraktionen nicht so Träge wie eine Swing-Anwendung verhält, sollte man SWT verwenden.
- der Komponenten von Windows wie zum Beispiel den Internet Explorer beinhaltet, sollte man SWT wählen.
- der innerhalb der Eclipse-Plattform als Plug-In realisiert wird, muss man SWT wählen.

Das Eclipse-Projekt hat mit SWT die Client-Entwicklung bereichert. In jedem Fall sollte man vor Projektbeginn prüfen, ob SWT nicht eine geeignetere Alternative darstellt.

REFERENZEN

- [1] Quellcode
  (/public/opensource/client-swing-swt/ClientSwingSWTBeispiel.zip)
- [2] Deploy an SWT application using Java Web Start
  (https://www.ibm.com/developerworks/java/library/os-jws/)
- [3] Comparing SWT and Swing
  (http://cld.blog-city.com/readblog.cfm?BID=15428)
- [4] High Performance GUIs with the JFC/Swing API
  (http://developer.java.sun.com/developer/community/chat/JavaLive/2003/jl0121.html)
- [5] SWT/JFace
  (http://eclipsewiki.editme.com/SWT)
- [6] SWT Component
  (http://www.eclipse.org/swt/)
- [7] Platform Plug-in Developer Guide
  (http://www.eclipse.org/documentation/)
- [8] Workbench User Guide
  (http://www.eclipse.org/documentation/)