Einführung

Borland®
Delphi™ 6
für Windows

Borland GmbH, Robert-Bosch-Straße 11, D-63225 Langen

Deutsche Ausgabe © 2001 Borland GmbH, Robert-Bosch-Straße 11, D-63225 Langen, Telefon 06103/979-0, Fax 06103/979-290

Update/Übertragung ins Deutsche: Krieger, Zander & Partner GmbH, München

Satz: Krieger, Zander & Partner GmbH, München

Hauptsitz: 100 Enterprise Way, Scotts Valley, CA 95066-3249, +1-(408)431-1000

Niederlassungen in: Australien, Deutschland, Frankreich, Großbritannien, Hong Kong, Japan, Kanada, Lateinamerika, Mexiko, den Niederlanden und Taiwan

HDE1360GE21000

Inhalt

Index

Kapitel

1

Einführung

Die vorliegende *Einführung* gibt einen Überblick über die Entwicklungsumgebung und die Funktionen von Delphi. Außerdem erfahren Sie hier, wo detaillierte Informationen über Delphi und die zahlreichen verfügbaren Tools zu finden sind.

In Kapitel 2, »Die Umgebung kennenlernen«, werden die wichtigsten Tools des Delphi-Desktops bzw. der integrierten Entwicklungsumgebung (IDE) erläutert. Kapitel 3, »Programmieren mit Delphi«, zeigt Ihnen, wie Sie mit einigen dieser Tools eine Anwendung erstellen. In Kapitel 4, »Die erste Anwendung: ein Texteditor«, lernen Sie in Form schrittweiser Anleitungen, wie Sie ein Programm für einen Texteditor schreiben. Kapitel 5 schließlich, »Die Umgebung anpassen«, beschreibt, wie Sie die Delphi-Entwicklungsumgebung an die speziellen Erfordernisse Ihres Entwicklungsprojekts anpassen.

Was ist Delphi?

Delphi ist eine objektorientierte, visuelle *RAD-Programmierumgebung* (Rapid Application Development, schnelle Entwicklung von Anwendungen), mit der Sie bei einem Minimum an manueller Programmierung hocheffiziente Anwendungen für Microsoft Windows 2000, Windows 98 und Windows NT erstellen können. Zudem ist Delphi die plattformübergreifende Entwicklungslösung, wenn es zusammen mit der von Borlands entwickelten Version für Linux eingesetzt wird. Delphi stellt alle Tools zur Verfügung, die Sie benötigen, um Anwendungen zu entwickeln, zu testen und zu vertreiben. Dazu gehören eine umfangreiche Bibliothek, wiederverwendbare Komponenten, eine Reihe von Entwurfs-Tools, Anwendungs- und Formularvorlagen sowie Programmierexperten.

Wo Sie Informationen finden

Für Delphi stehen vielerlei Informationsquellen zur Verfügung, die in diesem Kapitel beschrieben werden:

- Das Hilfesystem
- Die Dokumentation in Buchform
- Die Developer Support Services von Borland und die Borland-Website.

Informationen über die neuen Features dieser Programmversion finden Sie im Hilfesystem unter »Neuerungen« und auf der Website www.borland.de/delphi.

Hilfesystem

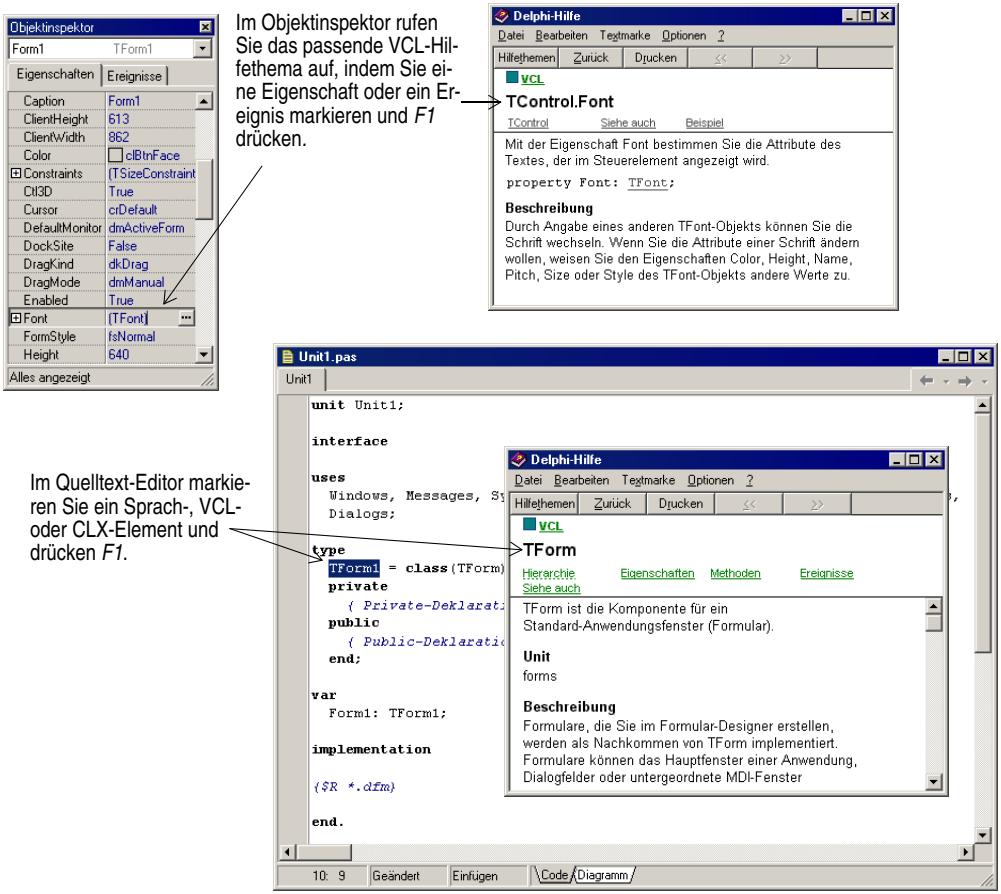
Im Hilfesystem von Delphi finden Sie ausführliche Informationen über die Leistungsmerkmale der Bedieneroberfläche, über die Sprachimplementierung, über Programmiertechniken sowie über die in der Visual Components Library (VCL) und der Borland Component Library for Cross Reference (CLX) enthaltenen Komponenten. Es umfaßt das gesamte Material aus dem Buch *Entwicklerhandbuch* und *Object Pascal Sprachreferenz* sowie eine Reihe von Hilfedateien zu anderen Delphi-Features.

Um das Inhaltsverzeichnis des Hilfesystems aufzurufen, wählen Sie *Hilfe / Delphi-Hilfe* oder *Hilfe / Hilfe zu Delphi-Tools* und öffnen die Registerkarte *Inhalt*. Um Beschreibungen zu VCL- oder CLX-Objekten oder einem sonstigen Thema nachzuschlagen, öffnen Sie die Registerkarte *Index* oder *Suchen* und geben den Suchbegriff ein.

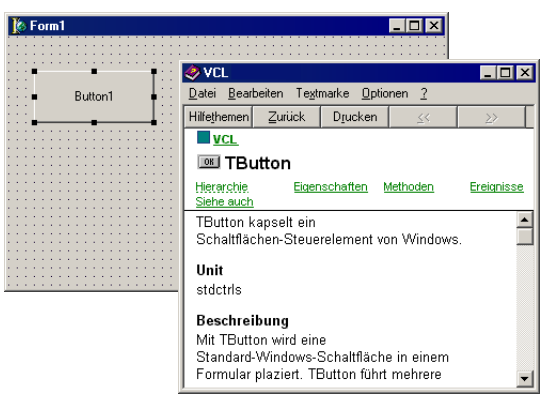
Kontextbezogene Hilfe

Zu VCL, CLX oder einem beliebigen Bereich der Entwicklungsumgebung, etwa Menüoptionen, Dialogfenster, Symbolleisten oder bestimmte Komponenten, können Sie

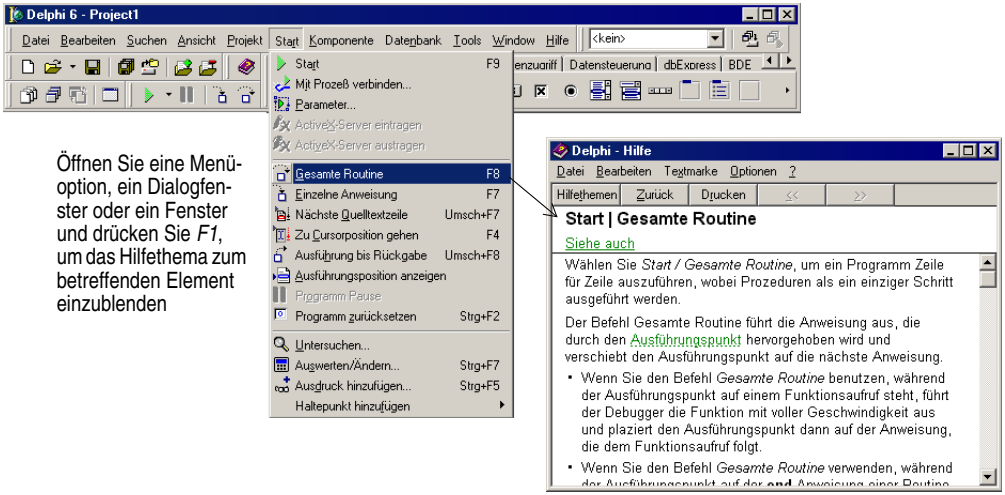
kontextbezogene Hilfethemen aufrufen. Dazu markieren Sie das betreffende Element und drücken *F1*.



Sie erhalten ebenfalls kontextbezogene Informationen, wenn Sie in einem Dialogfenster auf die Schaltfläche *Hilfe* klicken.



Fehlermeldungen vom Compiler oder Linker erscheinen in einem speziellen Fenster unter dem Quelltext-Editor. Um Hilfe zu Kompilierfehlern aufzurufen, markieren Sie in der Liste eine Meldung und drücken *F1*.



Typographische Konventionen

Für dieses Handbuch gelten die in Tabelle 1.1 aufgeführten typographischen Konventionen.

Tabelle 1.1 Schriftarten und Symbole in den Handbüchern

Schriftart/Symbol	Bedeutung
`Schreibmaschinenschrift`	Text in Schreibmaschinenschrift steht für Object Pascal-Quelltext oder für Text, wie er auf dem Bildschirm erscheint. Weiterhin kann es sich um Text handeln, den Sie in dieser Form eingeben müssen.
[ ]	Eckige Klammern im Text oder in Syntax-Listings kennzeichnen optionale Elemente. Bei der Eingabe dieser Elemente sind die Klammern wegzulassen.
Fettschrift	Fettschrift kennzeichnet reservierte Schlüsselwörter von Object Pascal oder Compiler-Optionen.
Kursiv	Kursiv geschriebene Wörter verweisen auf Object Pascal-Bezeichner, wie Variablen, Komponenten, Ereignisse, Methoden und Eigenschaften. Kursivschrift dient auch zur Hervorhebung bestimmter Wörter, etwa neuer Begriffe.
Tasten	Diese Schriftart bezeichnet eine Taste. Beispiel: "Drücken Sie *Esc*, um das Menü zu verlassen."

Borland Developer Support Services

Falls Sie Hilfe zu der Installation benötigen, bietet Borland zusätzliche Unterstützung unter der Nummer +49 (0) 180 5003065 an.

Telefonisch ist unser Support-Service unter der Nummer 0800 4677473 (Deutschland), 0800 552859 (Schweiz) und 0660 891 (Österreich) erreichbar. Weitere Informationen über Support finden Sie unter http://www.borland.com/bww/europe/devsupport.

Über die genannte Web Site haben Sie Zugang zu vielen Newsgroups, in denen Entwickler Informationen, Tips und Tricks austauschen. Hier finden Sie auch eine Liste mit Publikationen über Delphi.

Gedruckte Dokumentation

Die vorliegende *Einführung* gibt Ihnen einen ersten Überblick über Delphi. Wie Sie zusätliche Dokumentation bestellen können, erfahren Sie auf der Web Site von Borland (www.borland.de).

Kapitel

2

Die Umgebung kennenlernen

In diesem Kapitel erfahren Sie, wie Sie Delphi starten. Zudem bietet es eine Einführung in die wichtigsten Bestandteile und Tools des Desktops bzw. der integrierten Entwicklungsumgebung.

Delphi starten

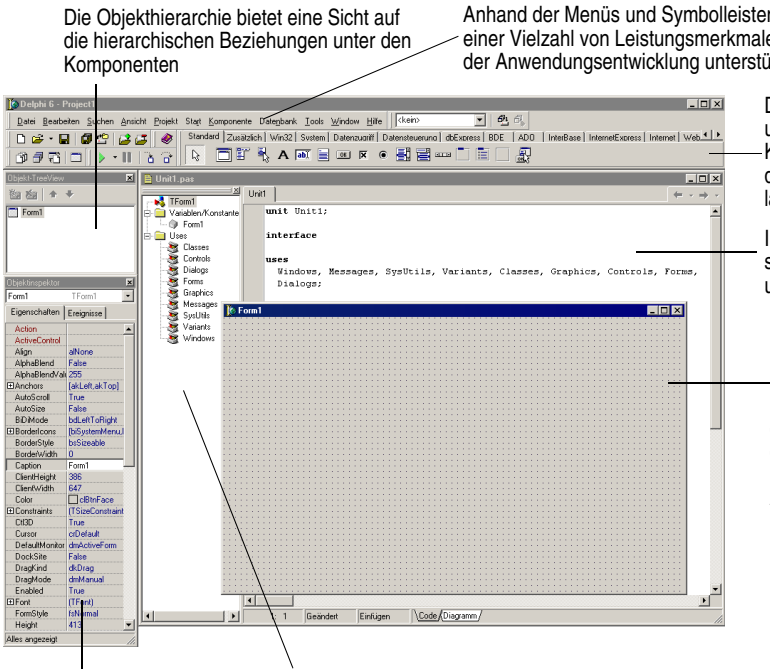
Um Delphi zu starten, gibt es mehrere Möglichkeiten:

- Doppelklicken Sie auf das Delphi-Symbol (sofern Sie eine Verknüpfung erstellt haben).
- Wählen Sie im Startmenü von Windows den Befehl *Programme / Borland Delphi 6 / Delphi 6.*
- Wählen Sie im Startmenü von Windows den Befehl *Ausführen,* und geben Sie dann *Delphi32* ein.
- Doppelklicken Sie im Verzeichnis DELPHI\BIN auf DELPHI32.EXE.

Die IDE

Wenn Sie Delphi zum ersten Mal starten, sehen Sie einige der wichtigsten Tools in der *IDE* (Integrated Development Environment, integrierte Entwicklungsumgebung). Zur Entwicklungsumgebung von Delphi gehören die Menüs und Symbolleisten, eine Komponentenpalette, der Objektinspektor, die Objekthierarchie (auch Objekt-Tree-View), ein Quelltext-Editor, ein Code-Explorer, die Projektverwaltung (auch Projekt-Manager) und viele andere Tools. Welche Leistungsmerkmale und Komponenten Ih-

nen im Speziellen zur Verfügung stehen, hängt davon ab, welche Delphi-Version Sie erworben haben.



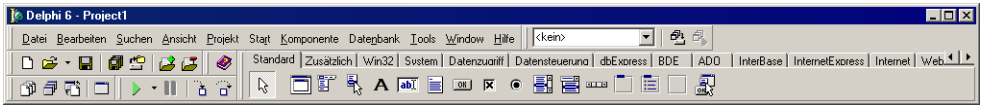
Das Entwicklungsmodell von Delphi basiert auf sogenannten *Two-way-Tools*. Das bedeutet, daß Sie leicht zwischen visuell orientierten und textbezogenen Programmierwerkzeugen wechseln können. Nachdem Sie beispielsweise Schaltflächen und andere grafische Elemente im Formular-Designer angeordnet haben, können Sie sich in der zugehörigen Formulardatei sofort die textuelle Entsprechung des Formulars ansehen. Und umgekehrt können Sie den von Delphi generierten Quelltext manuell bearbeiten, ohne dadurch den Zugriff auf die visuelle Programmierumgebung zu verlieren.

Innerhalb der IDE sind alle Programmier-Tools in bequemer Reichweite. Sie können grafische Benutzeroberflächen entwerfen, Klassenbibliotheken durchsuchen, Quelltext schreiben und Projekte compilieren, testen, debuggen und verwalten, ohne die IDE zu verlassen.

Wie Sie die IDE organisieren und konfigurieren, erfahren Sie in Kapitel 5, »Die Umgebung anpassen«.

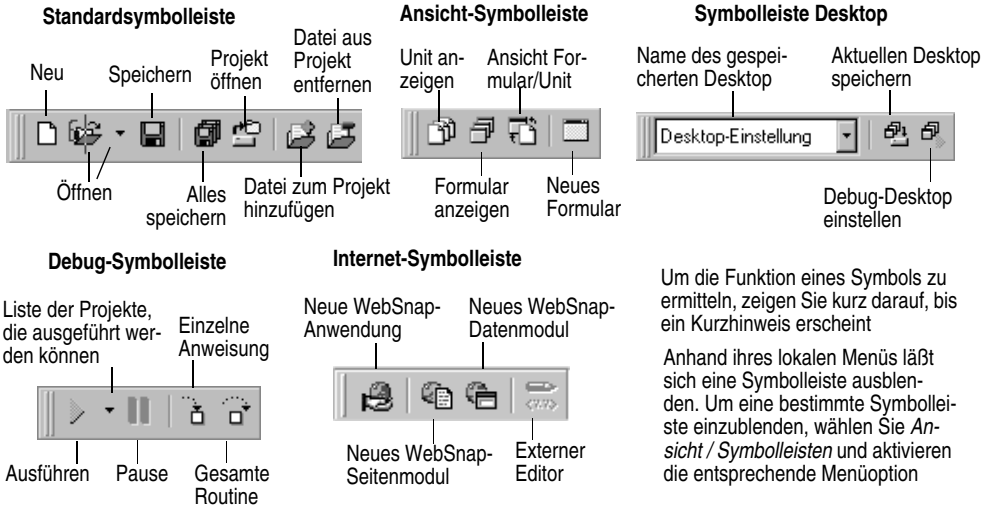
Die Menüs und Symbolleisten

Das Hauptfenster, das den oberen Bildschirmbereich einnimmt, enthält das Hauptmenü, die Symbolleisten und die Komponentenpalette.



Das Hauptfenster in der Standardanordnung.

Die Symbolleisten von Delphi ermöglichen einen raschen Zugriff auf die gebräuchlichsten Operationen. Jede dieser Operationen hat auch seine Entsprechung in einem Befehl des Menüsystems.



Zu vielen Operationen gibt es neben den Befehlssymbolen auch Tastenkürzel (Hotkeys). Ist ein solches Tastenkürzel verfügbar, so erscheint es im entsprechenden Menü der Menüleiste (rechts neben dem Optionsnamen).

Viele Tools und Symbole sind mit einem Menü verbunden, das kontextspezifische Befehle enthält. Menüs dieser Art werden als *lokale Menüs* bezeichnet. Sie lassen sich öffnen, indem Sie das betreffende Objekt mit der rechten Maustaste anklicken.

Jede Symbolleiste läßt sich an Ihre individuellen Bedürfnisse anpassen. So können Sie beliebige Befehle in sie integrieren oder ihre Position innerhalb des Fensters verändern. Erläuterungen hierzu finden Sie in den Abschnitten »Menüs und Symbolleisten anordnen« auf Seite 5-1 und »Desktop-Layouts speichern« auf Seite 5-5.

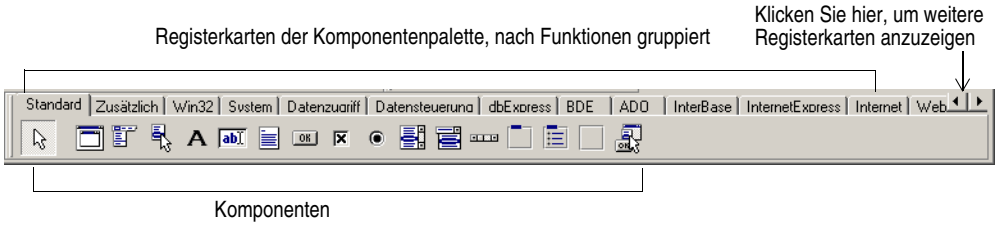
Weitere Informationen

Um das Hilfethema zu einer Menüoption aufzurufen, setzen Sie den Cursor darauf und drücken *F1*.

Die Komponentenpalette, der Formular-Designer und der Objektinspektor

Durch die Zusammenarbeit von Komponentenpalette, Formular-Designer, Objektinspektor und Objekthierarchie werden Sie bei der Entwicklung einer Benutzeroberfläche unterstützt.

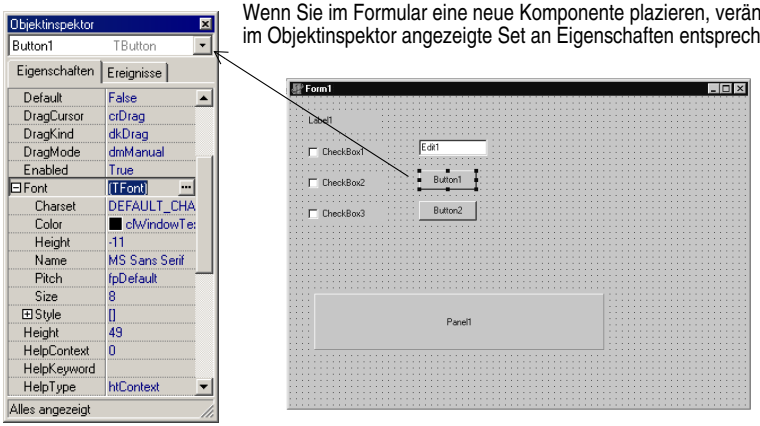
Die *Komponentenpalette* besteht aus Registerkarten mit Symbolgruppen, die visuelle und nichtvisuelle VCL- und CLX-Komponenten repräsentieren. Die einzelnen Karten untergliedern die Komponenten in verschiedene funktionale Gruppen. Die Registerkarten *Standard, Zusätzlich* und *Win32* etwa umfassen Steuerelemente wie Eingabefelder oder die Schaltflächen *Nach oben/Nach unten*. Und die Karte *Dialoge* stellt gebräuchliche Dialogfelder für Operationen wie das Öffnen oder Speichern von Dateien bereit.



Jede Komponente hat spezifische Attribute – Eigenschaften, Ereignisse und Methoden –, die die Steuerung der Anwendung erlauben.

Nachdem Sie in das Formular bzw. in den *Formular-Designer* Komponenten eingefügt haben, können Sie die Komponenten so plazieren, wie sie in der Bedieneroberfläche erscheinen sollen. Diese Komponenten können Sie anschließend mit Hilfe des *Objektinspektors* bearbeiten, indem Sie die Entwurfszeiteigenschaften festlegen, Ereignisbehandlungsroutinen erstellen oder sichtbare Eigenschaften und Ereignisse filtern: Erst dadurch stellen Sie die Verbindung zwischen dem äußeren Erscheinungsbild Ihrer Anwendung und dem Code her, der für die Funktionalität verantwortlich ist. Weitere

Erläuterungen hierzu finden Sie im Abschnitt »Komponenten in einem Formular plazieren« auf Seite 3-2.



Weitere Informationen

Suchen Sie im Hilfeindex nach dem Begriff »Komponentenpalette«.

Die Objekthierarchie

In der *Objekthierarchie* wird die strukturelle Hierarchie der Komponenten (*gleichrangig, übergeordnet* oder *untergeordnet*) in Form eines Baumdiagramms widergespiegelt. Dieses Baumdiagramm und der Objektinspektor und Formular-Designer sind synchron aufeinander abgestimmt, so daß bei Auswahl eines anderen Objekts in der Objekthierarchie sowohl im Objektinspektor als auch im Formular die jeweilige Entsprechung ausgewählt wird.

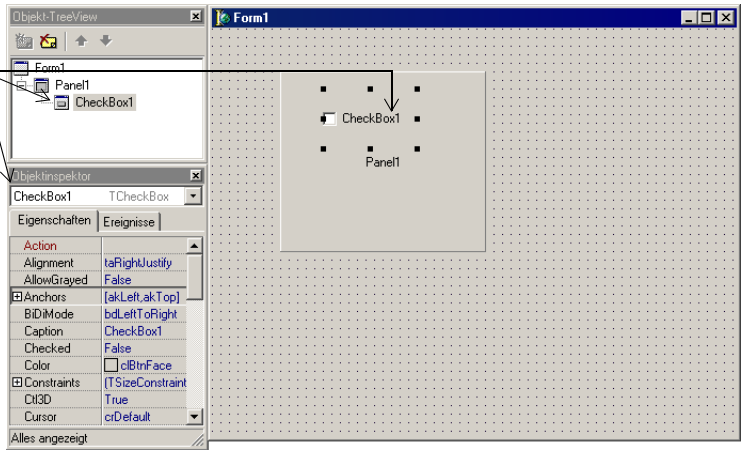
Mit Hilfe der Objekthierarchie (auch als *Objekt-TreeView* bezeichnet) läßt sich die Beziehung zwischen verwandten Komponenten ändern. Wenn Sie beispielsweise ein Bedienfeld und ein Kontrollfeld in das Formular einfügen, so sind die beiden Komponenten zunächst gleichrangig. Doch wenn Sie in der Objekthierarchie das Kontrollfeld auf das Bedienfeldsymbol ziehen, so wird das Kontrollfeld dem Bedienfeld untergeordnet.

Sind die Eigenschaften eines Objekts nicht vollständig definiert worden, so erscheint in der Objekthierarchie neben diesem Objekt ein rotes Fragezeichen. Sie haben auch die Möglichkeit, per Doppelklick auf ein Objekt im Baumdiagramm den Quelltext-Editor an einer Stelle zu öffnen, an der Sie eine Ereignisbehandlungsroutine schreiben können.

Wird die Objekthierarchie nicht angezeigt, wählen Sie *Ansicht / Objekt-TreeView.*

Objekthierarchie, Objektinspektor und Formular-Designer arbeiten synchron zusammen. Klicken Sie auf ein Objekt im Formular, so wird es automatisch auch in der Objekthierarchie und im Objektinspektor aktiviert und umgekehrt.

Drücken Sie *Alt+Umschalt+F11,* um die Objekthierarchie zu aktivieren.



Die Objekthierarchie eignet sich insbesondere zur Anzeige von Beziehungen zwischen Datenbankobjekten.

Weitere Informationen

Suchen Sie im Hilfeindex nach dem Begriff »Objekthierarchie«.

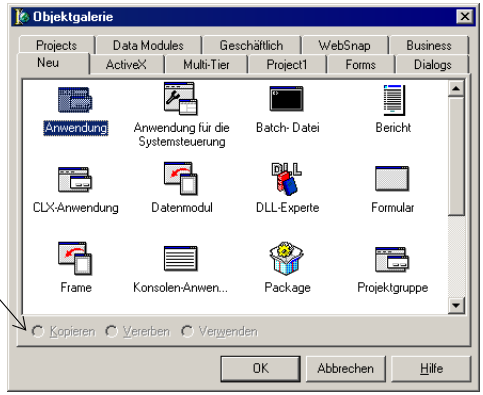
Die Objektablage

Die Objektablage enthält Formulare, Dialogfenster, Datenmodule, Experten, DLLs, Beispielanwendungen und andere Objekte, die den Entwicklungsvorgang vereinfachen. Um ein Projekt zu beginnen, wählen Sie *Datei / Neu / Weitere.* Das Dialogfenster *Objektgalerie* wird eingeblendet. Dieses Dialogfenster stellt die Objektablage dar. Schauen Sie nach, ob Sie in der Objektablage ein ähnliches Objekt finden wie das, das Sie erstellen wollen.

Auf den Registerkarten der Objektablage finden Sie Objekte wie Formulare, Rahmen und Units sowie Experten zum Erstellen besonderer Objekte

Sie können ein vorhandenes Objekt kopieren, vererben oder verwenden:

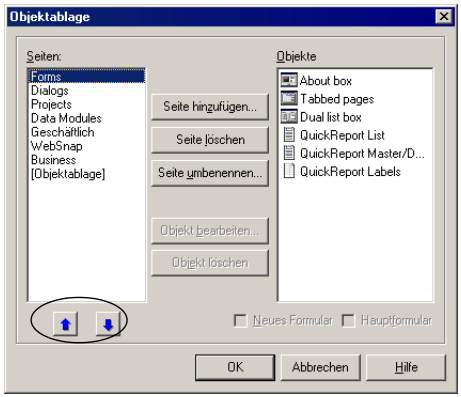
Kopieren (der Standard) legt in Ihrem Projekt eine Kopie des Objekts an. *Vererben* bedeutet, Änderungen am Objekt in der Objektablage werden von dem Objekt in Ihrem Projekt geerbt. *Verwenden* bedeutet, Änderungen am Objekt im Projekt werden von dem Objekt in der Objektablage geerbt.



Um Objekte aus der Objektablage zu bearbeiten oder zu entfernen, wählen Sie entweder *Tools / Objektablage* oder klicken mit der rechten Maustaste in das Dialogfenster *Objektgalerie* und wählen *Eigenschaften*.

Sie können die Registerkarten der Objektablage ergänzen, entfernen oder auch umbenennen.

Klicken Sie auf die Pfeile, um die Reihenfolge zu ändern, in der eine Registerkarte im Dialogfenster *Objektgalerie* erscheint.



Wie Sie Projekt- und Formularvorlagen in die Objektablage einfügen, erfahren Sie im Abschnitt »Vorlagen in die Objektablage einfügen« auf Seite 5-10.

Weitere Informationen

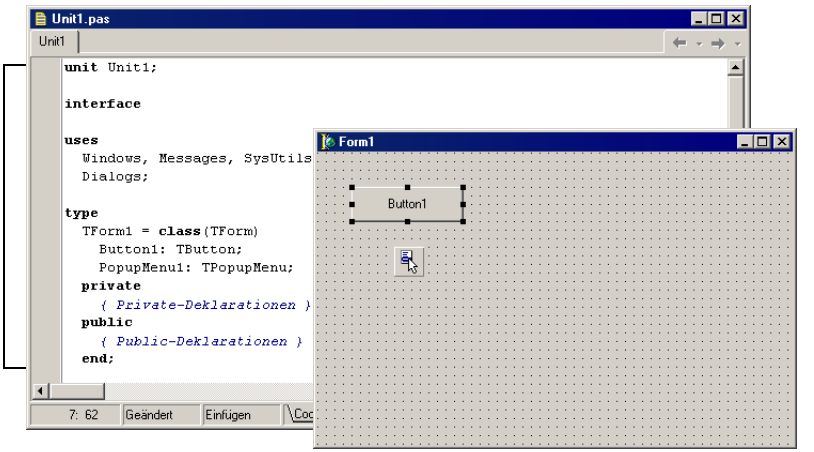
Suchen Sie im Hilfeindex nach dem Begriff »Objektablage«. Welche Objekte Ihnen bereits zur Verfügung stehen, hängt davon ab, welche Delphi-Version Sie erworben haben.

Der Quelltext-Editor

Beim Entwickeln einer Benutzeroberfläche erzeugt Delphi den zugrundeliegenden Objekt-Pascal-Code. Wenn Sie die Eigenschaften von Formularen und Objekten auswählen oder ändern, werden diese Änderungen automatisch in den Quelldateien widergespiegelt. Mit Hilfe des integrierten Quelltext-Editors – einem kompletten ASCII-Editor – können Sie Code selbst direkt in die Quelldateien eingeben.

Die ins Formular eingefügten Komponenten werden im Code widergespiegelt.

Automatisch erzeugter Code.



Delphi verfügt über verschiedene Hilfsmittel, die Sie bei der Code-Erstellung unterstützen, etwa die Programmierhilfen, die Code-Vervollständigung zur Vervollständigung von Klassen oder den Code-Browser.

Die Programmierhilfen

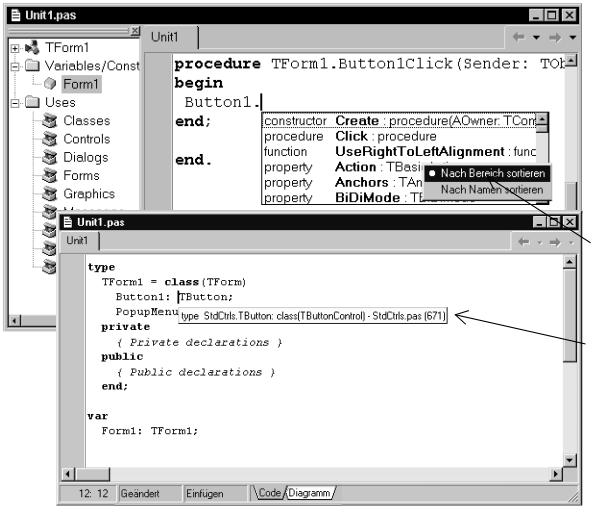
Im Quelltext-Editor stehen Ihnen verschiedene Programmierhilfen in Form kontextbezogener Popup-Fenster zur Verfügung.

Tabelle 2.1 Programmierhilfen

Programmhilfe	Funktionsweise
Code-Vervollständigung	Wenn Sie einen Klassennamen eingeben, dem ein Punkt folgt, werden in einer Liste alle Eigenschaften, Methoden und Ereignisse angezeigt, die für die betreffende Klasse zur Verfügung stehen. (Zur Auswahl markieren Sie einen Eintrag und drücken *Eingabe*.) Im Code-Abschnitt **interface** können Sie mehrere Elemente auswählen. Wenn Sie den Anfang eines Zuweisungs-Statements eingeben und *Strg+Leer* drücken, erscheint eine Liste der für eine Variable zulässigen Werte. Wenn Sie den Namen einer Prozedur, Funktion oder Methode eingeben, erscheint eine Liste mit Argumenten.
Code-Parameter	Wenn Sie den Namen einer Methode sowie eine öffnende runde Klammer eingeben, wird die Syntax der Argumente dieser Methode angezeigt.

Tabelle 2.1 Programmierhilfen (Fortsetzung)

Programmhilfe	Funktionsweise
Auswertung durch Kurzhinweis	Wenn während einer Debugger-Sitzung Ihr Programm angehalten wurde, können Sie den Mauszeiger auf eine beliebige Variable setzen, um sich deren aktuellen Wert anzeigen zu lassen.
Symbolinformation durch Kurzhinweis	Wenn Sie Ihren Quelltext bearbeiten, können sie den Mauszeiger auf einen beliebigen Bezeichner setzen, um sich dessen Deklaration anzeigen zu lassen.
Code-Vorlagen	Mit *Strg+J* läßt sich eine Liste mit gebräuchlichen Programmieranweisungen aufrufen, die Sie anschließend in Ihren Quelltext aufnehmen können. Zusätzlich zu den von Delphi bereitgestellten Code-Vorlagen können Sie auch eigene Vorlagen erstellen.



Dank der Code-Vervollständigung erscheint eine Liste mit den Eigenschaften, Methoden und Ereignissen einer Klasse, sobald Sie den Punkt in `Button1.` eingeben. Beim Eingeben wird die Listenauswahl automatisch gefiltert und auf die Elemente dieser Klasse reduziert. Um einen Listeneintrag in den Quelltext einzufügen, markieren Sie ihn und drücken *Return*.

Prozeduren, Eigenschaften und Funktionen sind farblich hervorgehoben.

Die Liste läßt sich alphabetisch sortieren, indem Sie mit der rechten Maustaste klicken und *Nach Namen sortieren* wählen.

Dank der Symbolinformation durch Kurzhinweis werden Deklarationsinformationen zu einem Bezeichner eingeblendet, wenn Sie den Mauszeiger darauf setzen.

Um die Programmierhilfen zu deaktivieren, wählen Sie *Tools / Editor-Optionen* und öffnen die Registerkarte *Programmierhilfe*. Aktivieren bzw. deaktivieren Sie die einzelnen Programmierhilfen im Abschnitt *Programmierhilfen*.

Vervollständigung von Klassen

Mittels der Vervollständigung von Klassen läßt sich Rumpfcode für Klassen generieren. Dazu setzen Sie den Cursor auf eine beliebige Stelle innerhalb einer Klassendeklaration des Abschnitts **interface** einer Unit und drücken entweder *Strg+Umschalt+C* oder öffnen das lokale Menü und wählen *Klasse beim Cursor vervollständigen*. Automatisch fügt Delphi zu den Deklarationen aller Eigenschaften, für die dies erforderlich ist, private **read-** und **write-**Bezeichner hinzu und erstellt dann den Rumpfcode für alle Methoden der Klasse. Diese Technik der Klassenvervollständigung gibt Ihnen auch die Möglichkeit, Klassendeklarationen für Methoden auszufüllen, die Sie bereits implementiert haben.

Um die Vervollständigung von Klassen zu aktivieren, wählen Sie *Tools / Umgebungsoptionen*, öffnen die Registerkarte *Explorer* und aktivieren die Option *Unvollst. Eigenschaften vervollst.*

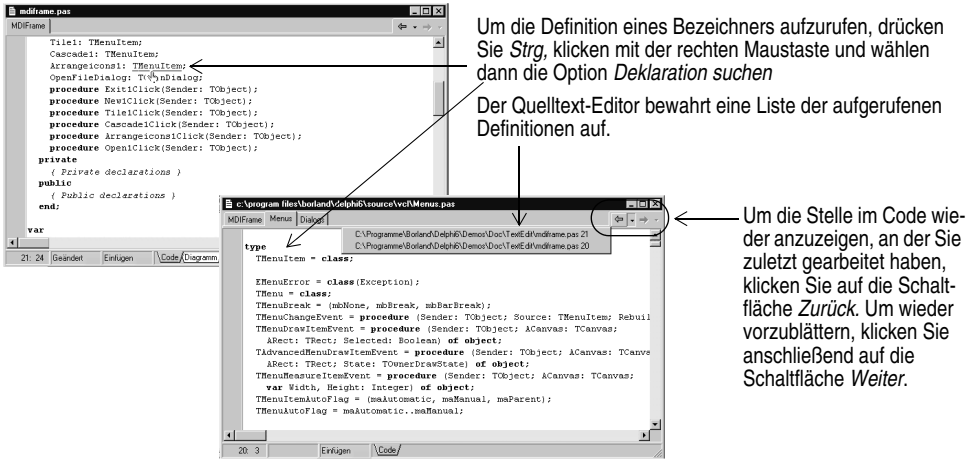
Weitere Informationen

Suchen Sie im Hilfeindex nach den Begriffen »Programmierhilfe« und »Vervollständigung von Klassen«.

Der Code-Browser

Wenn Sie den Mauszeiger auf den Namen einer Klasse, Variablen, Eigenschaft, Methode oder eines sonstigen Bezeichners setzen, wird Ihnen dank der Programmierhilfe *Symbolinformation durch Kurzhinweis* angezeigt, wo dieser Bezeichner deklariert worden ist. Drücken Sie *Strg*, so nimmt der Cursor die Form einer Hand an, der Bezeichner wird blau und unterstrichen angezeigt, und per Klick auf den Bezeichner wird seine Definition aufgerufen.

Entsprechend der Schaltflächen in einem Web-Browser besitzt der Quelltext-Editor Schaltflächen zum Vor- und Zurückblättern. Beim Aufrufen der Definition merkt sich der Quelltext-Editor, an welcher Stelle im Quelltext Sie jeweils waren. Und anhand der Dropdown-Liste neben den Schaltflächen *Weiter* und *Zurück* können Sie die verschiedenen Stationen dieser Referenzen erneut aufschlagen.



Sie können auch zwischen einer Prozedurdeklaration und ihrer Implementierung wechseln, indem Sie *Strg+Umschalt+*↑ oder *Strg+Umschalt+*↓ drücken.

Wie Sie Ihre Umgebung für die Code-Bearbeitung anpassen, erfahren Sie im Abschnitt »Den Quelltext-Editor anpassen« auf Seite 5-12.

Weitere Informationen

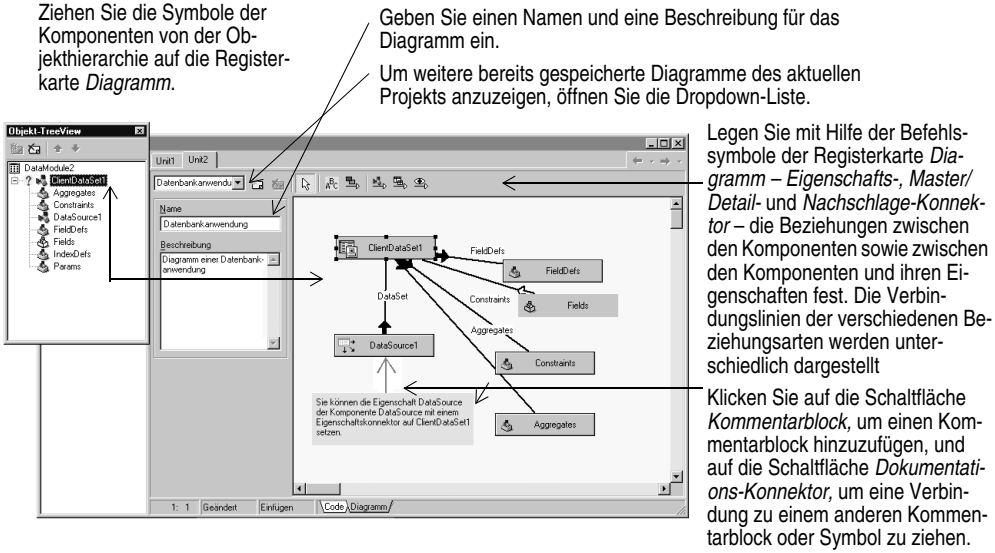
Suchen Sie im Hilfeindex nach dem Begriff »Quelltext-Editor«.

Die Registerkarte Diagramm

Im unteren Bereich des Quelltext-Editors sind ein oder auch mehrere Registerreiter zu sehen. Die Registerkarte *Code*, in die Sie den gesamten Quelltext eingeben, liegt standardmäßig vorne. Auf der Registerkarte *Diagramm* werden, je nach erworbener Delphi-Version, Symbole angezeigt sowie Verbindungslinien, die die Beziehung zwischen den Komponenten darstellen, die Sie in einem Formular oder Datenmodul plazieren. Solche Beziehungen sind die zwischen gleichrangigen Komponenten, zwischen über- und untergeordneten Komponenten und zwischen Komponenten und ihren Eigenschaften.

Um ein solches Diagramm zu erstellen, öffnen Sie die Registerkarte *Diagramm*. Um die Symbole vertikal anzuordnen, ziehen Sie sie einfach von der Objekthierarchie auf die Registerkarte *Diagramm*. Um sie horizontal anzuordnen, drücken Sie beim Ziehen die Taste *Umschalt*. Wenn Sie Symbole mit über- oder untergeordneten Komponenten oder mit einer Komponente-Eigenschaft-Abhängikeit auf die Registerkarte ziehen, werden die Linien, die die Abhängikeiten beschreiben, auch *Konnektoren* genannt, automatisch hinzugefügt. Angenommen, Sie fügen eine Datenmenge in ein Datenmodul ein und ziehen das Datenmengensymbol sowie seine Eigenschaftensymbole auf die Registerkarte *Diagramm*, so werden diese Symbole automatisch mit dem Datenmengensymbol verbunden.

Um für Komponenten ohne Abhängigkeiten Beziehungen zu definieren, können Sie mit Hilfe der Befehlssymbole oben auf der Registerkarte einen von vier Konnektortypen (*Dokumentations-*, *Eigenschafts-*, *Master/Detail-* und *Nachschlage-Konnektor*) hinzufügen. Es ist auch möglich, Kommentarblöcke hinzuzufügen, die miteinander oder mit einem entsprechenden Symbol verbunden sind.



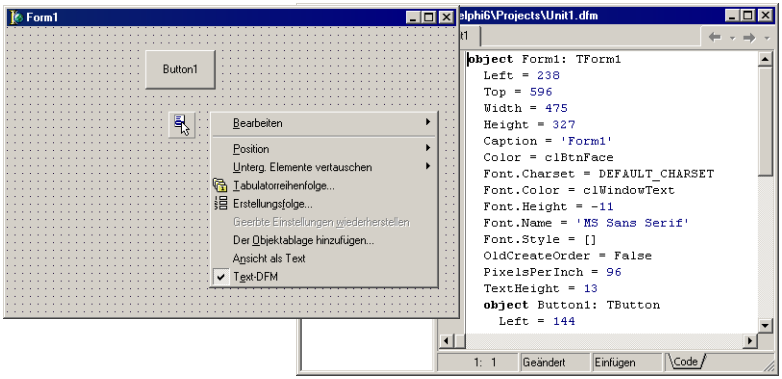
Sie können einen Namen und eine Beschreibung für das Diagramm eingeben, das Diagramm speichern und es nach Abschluß ausdrucken.

Weitere Informationen

Suchen Sie im Hilfeindex nach dem Begriff »Diagramm (Registerkarte)«.

Formularcode anzeigen

Formulare sind ein besonders sinnfälliger Teil der meisten Delphi-Projekte – denn darin entwerfen Sie die Benutzeroberfläche einer Anwendung. In der Regel entwerfen Sie die Formulare mit Hilfe der visuellen Tools von Delphi, und Delphi speichert die Formulare in Formulardateien. Formulardateien (.DFM bzw. .XFM bei einer CLX-Anwendung) beschreiben jede Komponente im Formular, einschließlich der Werte aller persistenten Eigenschaften. Um eine Formulardatei im Quelltext-Editor anzuzeigen und zu bearbeiten, klicken Sie mit der rechten Maustaste auf das Formular und wählen *Ansicht als Text*. Um die grafische Darstellung des Formulars wieder zu aktivieren, klicken Sie erneut mit der rechten Maustaste und wählen *Ansicht als Formular*.



Mit *Ansicht als Text* können Sie eine textuelle Beschreibung der Formularattribute im Quelltext-Editor anzeigen lassen.

Formulardateien lassen sich entweder im Textformat (der Standard) oder im Binärformat speichern. Um das Format neuer Formulare festzulegen, wählen Sie *Tools / Umgebungsoptionen*, öffnen die Registerkarte Designer und aktivieren bzw. deaktivieren die Option *Neue Formulare als Text*.

Weitere Informationen

Suchen Sie im Hilfeindex nach dem Begriff »Formulardateien«.

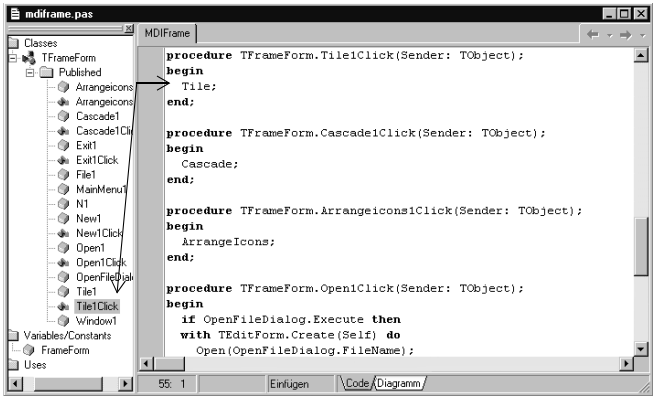
Der Code-Explorer

Wenn Sie Delphi öffnen, ist der Code-Explorer an der linken Seite des Quelltext-Editor-Fensters angedockt, vorausgesetzt der Code-Explorer ist in Ihrer Delphi-Version verfügbar. Der Code-Explorer zeigt ein Inhaltsverzeichnis des im Quelltext-Editor geöffneten Quelltexts in Form eines Baumdiagramms an, das die in Ihrer Unit definierten Typen, Klassen, Eigenschaften, Methoden, globalen Variablen und Routinen auflistet. Es zeigt zudem die anderen Units an, die in der **uses**-Klausel angeführt sind.

Über den Code-Explorer haben Sie auch die Möglichkeit, verschiedene Stellen im Quelltext-Editor anzusteuern. Wenn Sie z. B. im Code-Explorer auf eine Methode doppelklicken, so wird im Quelltext-Editor die Definition in der Klassendeklaration im **interface**-Abschnitt der Unit angezeigt.

Doppelklicken Sie auf einen Eintrag im Code-Explorer, so wird im Quelltext-Editor die Implementierung dieses Eintrags angezeigt. Drücken Sie *Strg+Umschalt+E*, um zwischen den beiden letzten Stellen im Code-Explorer und Quelltext-Editor zu wechseln

Jeder Eintrag im Code-Explorer besitzt ein Symbol, das seinen Typ bezeichnet.



Um einzustellen, wie der Inhalt des Code-Explorers angezeigt wird, wählen Sie *Tools / Umgebungsoptionen* und öffnen die Registerkarte *Explorer.* Weitere Erläuterungen hierzu finden Sie im Abschnitt »Den Code-Explorer anpassen« auf Seite 5-12.

Weitere Informationen

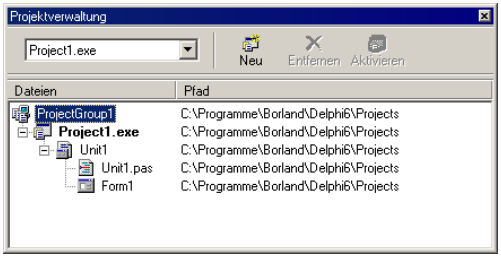
Suchen Sie im Hilfeindex nach dem Begriff »Code-Explorer«.

Die Projektverwaltung

Wenn Sie Delphi das erste Mal starten, wird automatisch ein neues Projekt erstellt. Ein Projekt umfaßt mehrere Dateien, aus denen sich die Anwendung bzw. die DLL, die Sie entwickeln, zusammensetzt. Diese Dateien – z. B. Formular-, Unit-, Ressourcen-, Objekt- oder Bibliotheksdateien – lassen sich in einem Projekt-Management-Tool, der sogenannten *Projektverwaltung,* anzeigen und organisieren. Um die Projektverwaltung anzuzeigen, wählen Sie *Ansicht / Projektverwaltung.*

Anhand der Projektverwaltung können Sie auch Informationen verwandter Projekte zu einer einzigen *Projektgruppe* zusammenbinden. Die Organisation verwandter Projekte in einer Gruppe hat den Vorteil, daß Sie mehrere Dateien, beispielsweise EXE-

Dateien, gleichzeitig compilieren können. Wie Sie Projektoptionen, etwa die Compilierung eines Projekts, ändern, erfahren Sie im Abschnitt »Projektoptionen festlegen« auf Seite 5-9.



Weitere Informationen

Suchen Sie im Hilfeindex nach dem Begriff »Projektverwaltung«.

Der Projekt-Browser

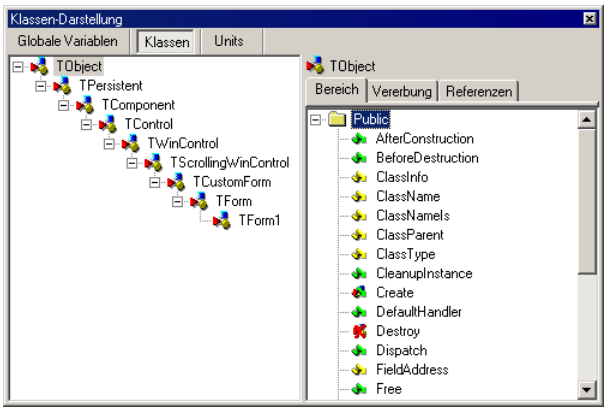
Der Projekt-Browser untersucht detaillierte Aspekte eines Projekts. Der Browser zeigt Klassen, Units und globale Symbole (Typen, Eigenschaften, Methoden, Variablen und Routinen) an, die für ein Projekt in einem Baumdiagramm deklariert worden sind. Um den Projekt-Browser zu öffnen, wählen Sie *Ansicht / Browser.*

Der Projekt-Browser besteht aus zwei Bereichen, die sich in ihrer Größe verändern lassen: Links der Inspektor und rechts die Details. Der Inspektor besitzt drei Registerkarten für globale Variablen, Klassen und Units

Auf der Registerkarte *Global* werden Klassen, Typen, Eigenschaften, Methoden, Variablen und Routinen angezeigt

Auf der Registerkarte *Klassen* erscheinen die Klassen in einem hierarchischen Diagramm

Auf der Registerkarte *Units* finden sich die in jeder Unit deklarierten Bezeichner sowie die anderen Units, die die einzelnen Units benutzen bzw. von diesen benutzt werden.



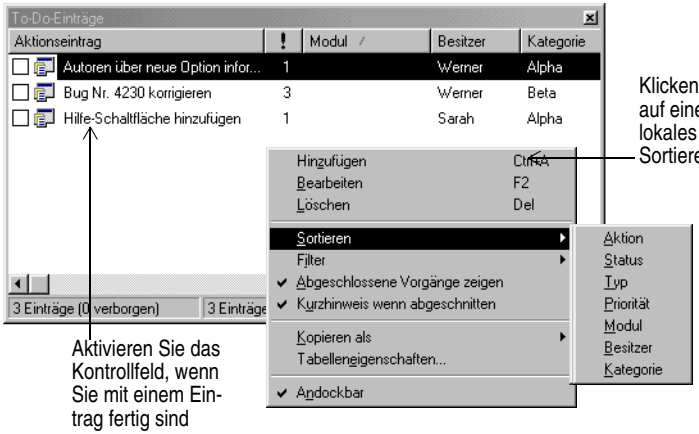
Standardmäßig zeigt der Projekt-Browser nur die Symbole der Units im aktuellen Projekt an. Sie können jedoch den Anzeigebereich so erweitern, daß alle in Delphi verfügbaren Symbole angezeigt werden. Wählen Sie dazu *Tools / Umgebungsoptionen*, und aktivieren Sie auf der Registerkarte *Explorer* die Option *Alle Symbole (inkl. VCL).*

Weitere Informationen

Suchen Sie im Hilfeindex nach dem Begriff »Projekt-Browser«.

To-Do-Listen

To-Do-Listen zeichnen Einträge auf, die für ein Projekt fertiggestellt werden müssen. Projektweite Einträge nehmen Sie in eine Liste auf, indem Sie sie einfach direkt in die Liste einfügen. Sie haben aber auch die Möglichkeit, spezifische Einträge direkt in den Quelltext einzufügen. Um die zu einem Projekt gehörigen Informationen anzuzeigen oder neue Informationen aufzunehmen, wählen Sie *Ansicht / To-Do-Liste*.



Klicken Sie mit der rechten Maustaste auf eine To-Do-Liste, so erscheint ein lokales Menü mit Befehlen, die dem Sortieren und Filtern der Liste dienen.

Aktivieren Sie das Kontrollfeld, wenn Sie mit einem Eintrag fertig sind

Weitere Informationen

Suchen Sie im Hilfeindex nach dem Begriff »To-Do-Listen«.

Kapitel

3

Programmieren mit Delphi

Die nachfolgenden Abschnitte bieten einen Überblick über die Software-Entwicklung mit Delphi – von der Projekterstellung, dem Einsatz von Formularen, der Codeerstellung über die Compilierung, Fehleranalyse, Distribution bis hin zur Erstellung unterschiedlicher internationaler Versionen einer Anwendung und der für die Entwicklung möglichen Projektarten.

Ein Projekt erstellen

Ein Projekt ist eine Sammlung an Dateien, die entweder zur Entwurfszeit erstellt oder beim Compilieren des Projektquelltexts automatisch erzeugt werden. Wenn Sie Delphi das erste Mal starten, wird ein neues Projekt angelegt. Dabei werden u. a. eine Projektdatei (PROJECT1.DPR), eine Unit-Datei (UNIT1.PAS) und eine Ressourcendatei (UNIT1.DFM bzw. UNIT1.XFM für CLX-Anwendungen) angelegt.

Ist bereits ein Projekt geöffnet und Sie möchten ein neues anlegen, wählen Sie entweder *Datei / Neu / Anwendung* oder *Datei / Neu / Weitere* und doppelklicken auf das Symbol *Anwendung*. Mit *Datei / Neu / Weitere* wird zunächst die Objektablage geöffnet, die weitere Formulare, Module und Frames bereitstellt, sowie vordefinierte Vorlagen (z. B. Dialogfenster), die Sie ins Projekt einbinden können. Die Objektablage ist im Abschnitt »Die Objektablage« auf Seite 2-6 ausführlich beschrieben.

Wenn Sie ein Projekt beginnen, sollten Sie wissen, was Sie entwickeln wollen, etwa eine Anwendung oder eine DLL. Welche Projektarten Sie mit Delphi entwickeln können, erfahren Sie im Abschnitt »Projektarten« auf Seite 3-10.

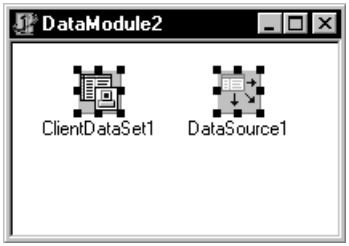
Weitere Informationen

Suchen Sie im Hilfeindex nach dem Begriff »Projekte«.

Datenmodule ins Projekt aufnehmen

Ein Datenmodul ist ein Formular, das ausschließlich Komponenten nichtvisueller Art enthält. Es ist durchaus möglich, solche unsichtbaren Komponenten zusammen mit visuellen in ein gewöhnliches Formular einzusetzen. Doch wenn Sie vorhaben, Gruppen mit Datenbank- und Systemobjekten mehrfach einzusetzen oder wenn Sie die Bereiche Ihrer Anwendung isolieren wollen, die für Datenbank-Connectivity und Business Rules verantwortlich sind, so läßt sich dies mit Hilfe von Datenmodulen recht praktisch organisieren.

Um ein Datenmodul zu erstellen, wählen Sie *Datei / Neu / Datenmodul*. Ein leeres Datenmodul wird geöffnet, das im Code-Editor in Form einer zusätzlichen Unit-Datei erscheint, und als neue Unit in das aktuelle Projekt aufgenommen wird. Das Einfügen nichtvisueller Komponenten in ein Datenmodul funktioniert genauso wie das Einfügen von Komponenten in ein Formular.



Um eine nichtvisuelle Komponente ins Datenmodul einzufügen, doppelklicken Sie in der Komponentenpalette auf eine nichtvisuelle Komponente

Wenn Sie ein vorhandenes Datenmodul erneut öffnen, werden seine Komponenten automatisch angezeigt.

Weitere Informationen

Suchen Sie im Hilfeindex nach dem Begriff »Datenmodule«.

Die Benutzeroberfläche erstellen

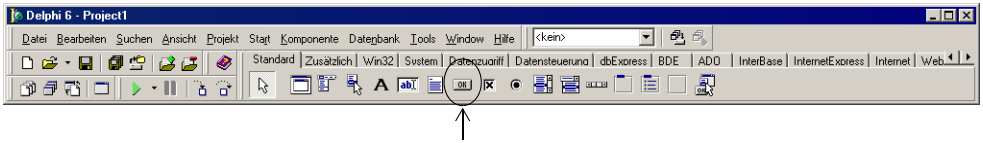
In Delphi erstellen Sie zunächst eine UI (User Interface, Benutzeroberfläche), indem Sie Komponenten aus der Komponentenpalette auswählen und ins Hauptformular einfügen.

Komponenten in einem Formular plazieren

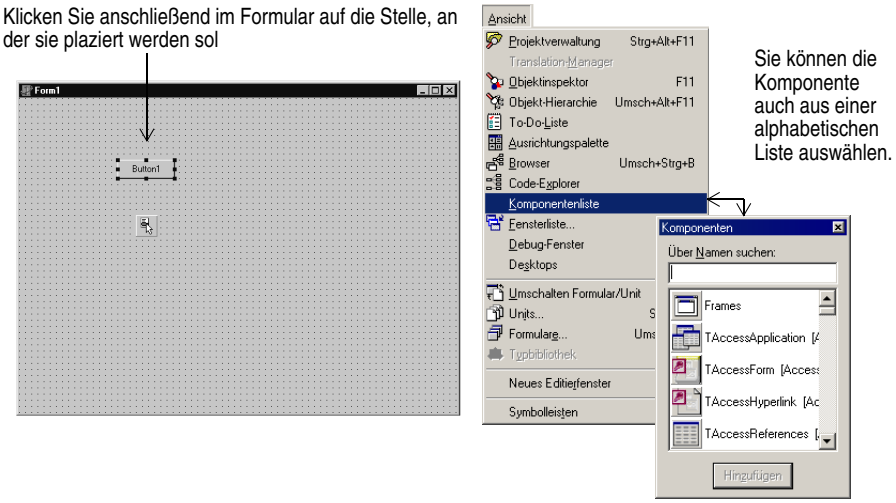
Um Komponenten in einem Formular zu plazieren, haben Sie folgende Möglichkeiten:

1 Sie doppelklicken auf die Komponente.

2 Sie wählen die Komponente auf der Palette aus, und klicken dann im Formular auf die Stelle, an der sie erscheinen soll.



Sie wählen die Komponente auf der Palette aus und ziehen sie auf die Stelle im Formular, an der sie erscheinen soll.



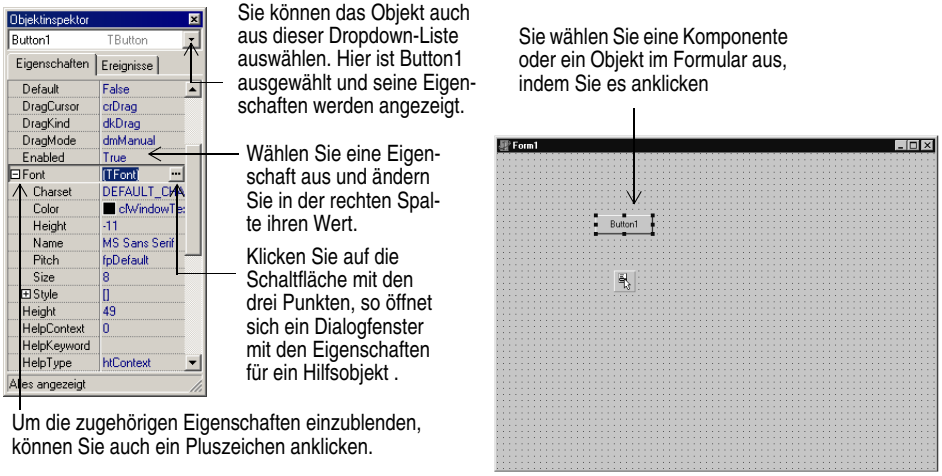
Weitere Informationen

Suchen Sie im Hilfeindex nach dem Begriff »Komponentenpalette«.

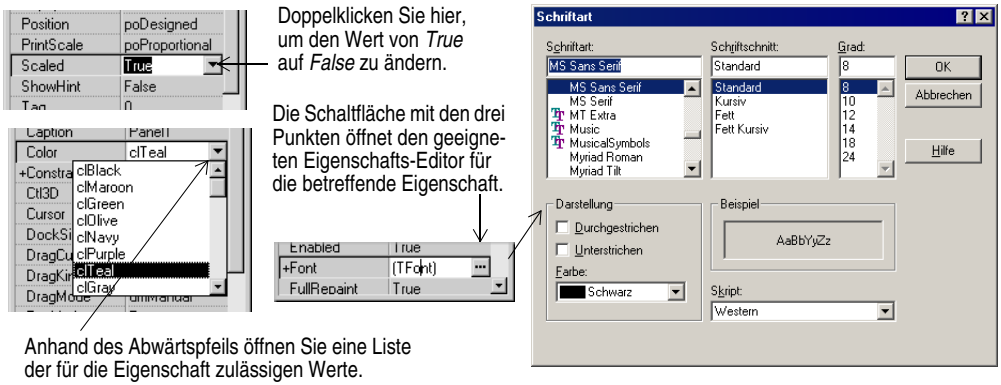
Komponenteneigenschaften festlegen

Nachdem Sie ein Formular mit Komponenten bestückt haben, legen Sie deren Eigenschaften fest und programmieren ihre Ereignisbehandlungsroutinen. Die Eigenschaften einer Komponente bestimmen ihr Aussehen und Verhalten in der Anwendung.

Wenn eine Komponente im Formular ausgewählt ist, werden ihre Eigenschaften im Objektinspektor angezeigt.



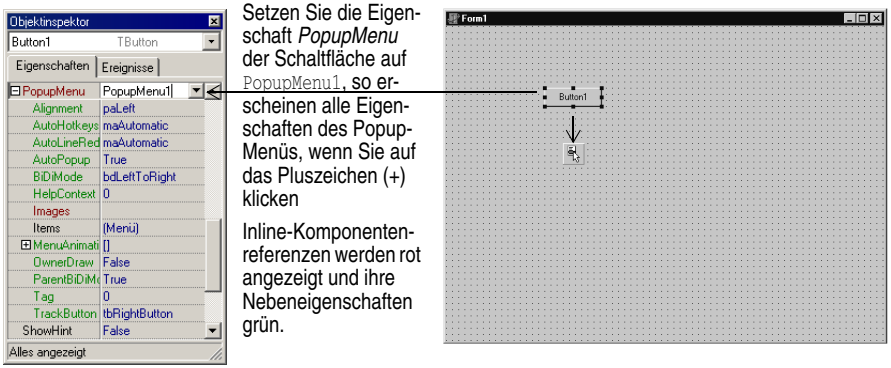
Viele Eigenschaften besitzen einfache Werte – etwa die Namen von Farben, logische Werte (*True* und *False*) oder Integerwerte. Bei logischen Werten können Sie per Doppelklick zwischen *True* und *False* umschalten. Für die Einstellung komplexerer Werte sind manche Eigenschaften mit Eigenschafts-Editoren verbunden. Neben solchen Eigenschaften finden Sie die Schaltfläche mit den drei Punkten. Bei anderen Eigenschaften hingegen, etwa einer Größe, tragen Sie einfach eine Zahl ein.



Werden im Formular mehrere Komponenten markiert, so erscheinen im Objektinspektor nur die Eigenschaften, die in allen ausgewählten Komponenten vorkommen.

Der Objektinspektor unterstützt auch ein- und ausblendbare *Inline-Komponentenreferenzen*. Dadurch haben Sie Zugriff auf die Eigenschaften und Ereignisse einer referenzierten Komponente, ohne die Komponente auswählen zu müssen. Wenn Sie beispielsweise eine Schaltfläche und ein Popup-Menü ins Formular einsetzen und die Schaltfläche auswählen, können Sie im Objektinspektor die Eigenschaft *PopupMenu*

auf PopupMenu1 setzen, worauf alle Eigenschaften des Popup-Menüs angezeigt werden.



Weitere Informationen

Suchen Sie im Hilfeindex nach dem Begriff »Objektinspektor«.

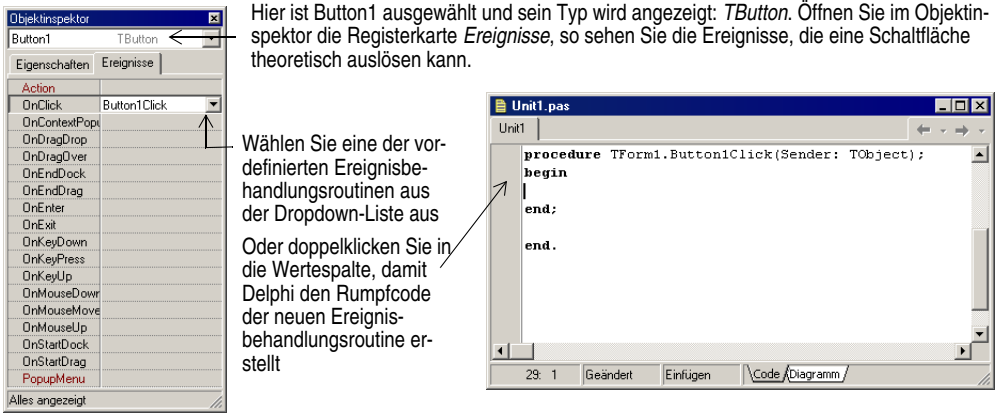
Code schreiben

Ein integraler Bestandteil einer jeden Anwendung ist der Code hinter den einzelnen Komponenten. Zwar stellt die RAD-Umgebung von Delphi bereits die meisten Erstellungsblöcke bereit, etwa die vorinstallierten visuellen und nichtvisuellen Komponenten, doch werden Sie Ereignisbehandlungsroutinen, Methoden und vielleicht einige eigene Klassen selbst schreiben müssen. Auch bei dieser Aufgabe werden Sie von der Entwicklungsumgebung unterstützt, und zwar durch Tausende bereitgestellter Objekte in den VCL- und CLX-Klassenbibliotheken. Wie Sie Ihren Quelltext bearbeiten, erfahren Sie im Abschnitt »Der Quelltext-Editor« auf Seite 2-8.

Ereignisbehandlungsroutinen schreiben

Oft muß der Code auf Ereignisse reagieren, die einer Komponente während der Laufzeit widerfahren. Ein Ereignis ist eine Verknüpfung zwischen etwas, das im System passiert, wie etwa ein Klick auf eine Schaltfläche, und einem Stück Quelltext, der auf dieses Geschehnis reagiert. Diesen Quelltext nennt man eine *Ereignisbehandlungsroutine.* Der Code einer Ereignisbehandlungsroutine ändert zum Beispiel Eigenschaftswerte oder ruft Methoden auf.

Um einige vordefinierte Ereignisbehandlungsroutinen zu einer Komponente im Formular anzuzeigen, wählen Sie die Komponente aus und öffnen im Objektinspektor die Registerkarte *Ereignisse*.



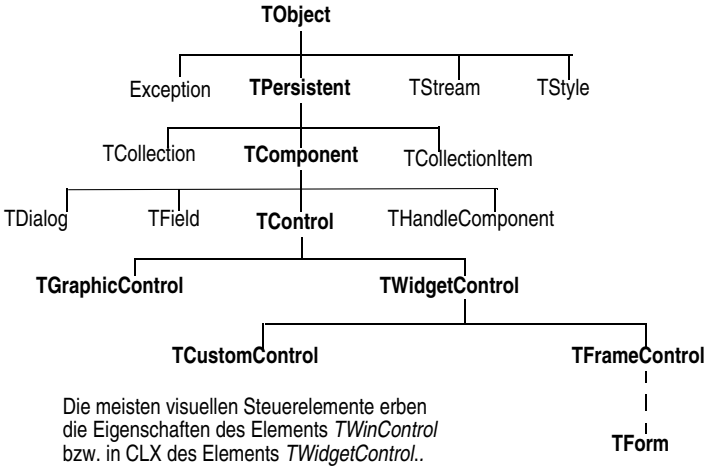
Weitere Informationen

Suchen Sie im Hilfeindex nach dem Begriff »Ereignisse«.

Die VCL- und CLX-Bibliotheken

Delphi wird mit zwei Klassenbibliotheken ausgeliefert, zu deren Objekten auch Komponenten oder Steuerelemente gehören, die Sie beim Schreiben von Code verwenden. Die eine, Visual Component Library (VCL), ist für die Erstellung von Windows-Anwendungen konzipiert, und die Borland Component Library for Cross Platform (CLX) für die Erstellung von Linux-Anwendungen. Beide Bibliotheken umfassen sowohl Objekte, die während der Laufzeit sichtbar sind, – also Editierfelder, Schaltflächen oder andere Elemente der Benutzerschnittstelle – als auch unsichtbare

(nichtvisuelle) Steuerelemente wie Datenmengen und Timer. Die folgende Abbildung zeigt einige der wichtigsten Klassen der VCL auf; die CLX-Hierarchie ist ähnlich.



Die von *TComponent* abgeleiteten Objekte verfügen über Eigenschaften und Methoden, die es ermöglichen, sie in der Komponentenpalette zu installieren und in Delphi-Formulare und Datenmodule einzufügen. Da die VCL- und CLX-Komponenten fest in die IDE integriert sind, können Sie für eine schnelle Anwendungsentwicklung Tools wie den Formular-Designer benutzen.

Komponenten verfügen über ein Höchstmaß an Kapselung. Schaltflächen sind beispielsweise so vorprogrammiert, daß sie durch ihre integrierte *OnClick*-Ereignisbehandlungsroutine auf Mausklicks reagieren. Wenn Sie eine Schaltfläche aus der VCL oder CLX verwenden, erübrigt es sich also, für den Fall, daß diese Schaltfläche angeklickt wird, Code zur Behandlung der generierten Ereignisse zu schreiben. Sie selbst brauchen nur noch die Routine schreiben, die beim Anklicken der Schaltfläche aufgerufen werden soll.

Die meisten Delphi-Versionen werden mit dem kompletten VCL- und CLX-Quellcode sowie mit Beispielen für die Objekt-Pascal-Programmierung ausgeliefert.

Weitere Informationen

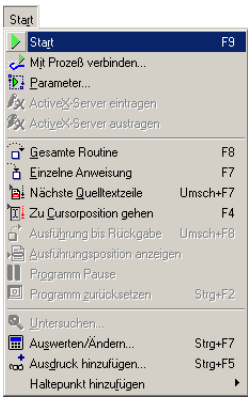
Suchen Sie im Inhaltsverzeichnis des Hilfesystem nach den Themen »VCL-Referenz« und »CLX-Referenz und im Hilfeindex nach dem Begriff »VCL«. Auf der Website www.borland.de/delphi finden Sie Informationen zur den Open-Source- oder Lizenzierungsmöglichkeiten bei der CLX-Programmierung.

Projekte compilieren und Fehlersuche

Nachdem Sie Ihren Code geschrieben haben, müssen Sie ihn compilieren und eventuelle Konflikte lösen. In Delphi können Sie entweder erst das Projekt compilieren und danach die Fehlersuche gesondert durchführen, oder Sie können unter Verwendung des integrierten Debuggers Compilierung und Fehlersuche in einem Arbeitsschritt ausführen. Um Ihr Programm mit Debug-Informationen zu compilieren, wählen Sie *Projekt / Optionen*, öffnen dann die Registerkarte *Compiler* und aktivieren die Option *Debug-Informationen.*

Delphi verwendet einen integrierten Debugger, so daß Sie die Programmausführung steuern, Variablen überwachen und während des Programmablaufs Datenwerte verändern können. Sie können den Code Zeile für Zeile ausführen lassen und so den Zustand des Programms an jedem Haltepunkt überprüfen. Um den integrierten Debugger zu verwenden, wählen Sie *Tools / Debugger-Optionen*, öffnen die Registerkarte *Allgemein* und aktivieren die Option *Integrierte Fehlersuche.*

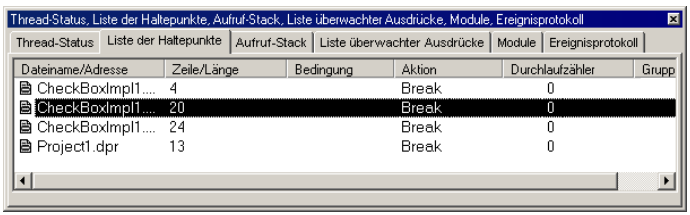
Sie beginnen eine Debug-Sitzung in der IDE, indem Sie auf der Debug-Symbolleiste auf die Schaltfläche *Start* klicken, in der Menüleiste *Start / Start* wählen oder *F9* drükken.



Wählen Sie im Menü *Start* einen der Debugging-Befehle aus. Einige der Befehle sind auch auf der Symbolleiste verfügbar.

Die Schaltfläche *Start*

Zu den zahlreichen Fenstern des Debuggers gehören die Fenster *Haltepunkte, Aufruf-Stack, Überwachte Ausdrücke, Lokale Variablen, Threads, Module, CPU* und *Ereignisprotokoll.* Sie öffnen diese Fenster mit *Ansicht / Debug-Fenster.* Welche Debugger-Ansichten verfügbar sind, hängt von Ihrer Delphi-Version ab.



Für eine einfachere Handhabung lassen sich die Debugging-Fenster kombinieren.

Wie Sie Debugging-Fenster kombinieren, erfahren Sie im Abschnitt »Tool-Fenster andocken« auf Seite 5-2.

Wenn Sie Ihren Desktop so eingerichtet haben, wie Sie es für Debug-Sitzungen wünschen, können Sie diese Einstellungen als Debug- oder Laufzeit-Desktop speichern. Bei allen künftigen Debug-Sitzungen wird dann genau dieses Desktop-Layout verwendet. Weitere Erläuterungen hierzu finden Sie im Abschnitt »Desktop-Layouts speichern« auf Seite 5-5.

Weitere Informationen

Suchen Sie im Hilfeindex nach den Begriffen »Debugging«, »Fehlersuche« und »Integrierter Debugger«.

Distribution Ihrer Anwendungen

Sie haben die Möglichkeit, Ihre Anwendungen andern zur Verfügung zu stellen, indem Sie sie vertreiben. Bei der Distribution einer Anwendung müssen Sie dafür sorgen, daß die Benutzer alle erforderlichen Dateien erhalten, einschließlich der EXE-Dateien, DLLs, Package-Dateien und Hilfsanwendungen. Zum Lieferumfang von Delphi gehört das Setup-Toolkit InstallShield Express, anhand dessen Sie aus diesen Dateien ein Installationsprogramm zusammenstellen können.

Weitere Informationen

Suchen Sie im Hilfeindex nach dem Begriff »Anwendungen vertreiben«.

Verschiedene Sprachversionen erstellen

Delphi bietet eine Reihe von Features, mit denen Sie Ihre Anwendungen an unterschiedliche Sprachräume anpassen können. Über die IDE und die VCL werden Unterstützung für Input-Method-Editoren (IMEs) und erweiterte Zeichensätze zur Verfügung gestellt, um die Lokalisierung Ihrer Projekte zu erleichtern. Die in einigen Versionen von Delphi verfügbare Translation Suite besteht aus Tools für die Software-Lokalisierung und die gleichzeitige Erstellung unterschiedlicher internationaler Versionen. Sie gibt Ihnen die Möglichkeit, mehrere lokalisierte Versionen als Teil eines einzigen Projekts zu verwalten.

Die Translation Suite umfaßt drei integrierte Tools:

- Der *Ressourcen-DLL-Experte* ist ein DLL-Experte zur Generierung und Verwaltung von Ressourcen-DLLs.
- Der *Translation Manager* ist ein Tabellengitter für die Anzeige und Bearbeitung der übersetzten Ressourcen.
- Das *Übersetzungswörterbuch* ist eine gemeinsam nutzbare Datenbank für die Speicherung von Übersetzungen.

Um den Ressourcen-DLL-Experten zu öffnen, wählen Sie *Datei / Neu / Weitere* und doppelklicken auf das Symbol *Ressourcen-DLL-Experte*. Um die Übersetzungs-Tools zu konfigurieren, wählen Sie *Tools / Optionen für Übersetzungs-Tools*.

Weitere Informationen

Suchen Sie im Hilfeindex nach dem Begriff »Internationale Anwendungen«.

Projektarten

Alle Delphi-Versionen unterstützen die 32-Bit-Windows-Programmierung, DLLs, Packages, selbstdefinierbare Komponenten, Multithreading, COM (Component Object Model) und Automatisierungs-Controller sowie Multiprozeß-Debugging. Einige Versionen unterstützen auch Server-Anwendungen wie etwa Webserver-Anwendungen, Datenbank-Anwendungen, COM-Server, Multi-Tier-Anwendungen, CORBA und Entscheidungsunterstützungssysteme.

Weitere Informationen

Wenn Sie nicht genau wissen, welche Tools Ihre Delphi-Version unterstützt, sollten Sie die Feature-Liste auf der Website www.borland.de/delphi zu Rate ziehen.

CLX-Anwendungen

In Delphi können Sie plattformübergreifende Anwendungen entwickeln, die Sie auf die Linux-Version von Delphi portieren können, wo Sie Ihr Projekt für die Ausführung auf Linux compilieren, debuggen und vertreiben können. Um eine CLX-Anwendung zu entwickeln, wählen Sie *Datei / Neu / CLX-Anwendung*. Die IDE ist ähnlich wie die einer regulären Delphi-Anwendung, nur erscheinen auf der Komponentenpalette und in der Objektablage eben die Komponenten und Elemente, die Sie in einer CLX-Anwendung verwenden können. Windows-spezifische Features, die von Delphi unterstützt werden, werden auf Linux-Umgebungen nicht direkt portiert.

Weitere Informationen

Um herauszufinden, welche Komponenten sich für die Entwicklung plattformübergreifender Anwendungen eignen, sollten Sie über das Inhaltsverzeichnis des Hilfesystems das Thema »CLX-Referenz« aufschlagen.

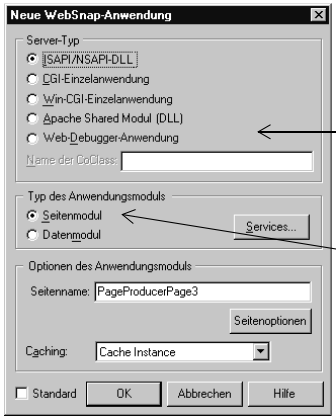
Web-Server-Anwendungen

Eine Web-Server-Anwendung arbeitet mit einem Web-Server zusammen, um eine Client-Anfrage zu verarbeiten und eine HTTP-Meldung in Form einer Webseite zurückzusenden. Für die Publikation von Daten auf dem Web verfügt Delphi je nach Version über zwei unterschiedliche Technologien.

Um eine grundlegende Web-Server-Anwendungen zu entwickeln, erstellen Sie ein Webmodul für das Weiterleiten von Anfragen, die Definition von Aktionen, das Erstellen von HTML-Seiten und das Schreiben von Ereignisbehandlungsroutinen für Windows- und Linux-Anwendungen. Um eine WebBroker-Web-Server-Anwendung zu erstellen, wählen Sie *Datei / Neu / Weitere* und doppelklicken auf das Symbol *Web-Server-Anwendung*. Aus den Registerkarten *Internet* und *InternetExpress* der Komponentenpalette können Sie Komponenten in Ihr Webmodul einfügen.

WebSnap erweitert diese Funktionalität durch Adapter, zusätzliche Dispatcher, zusätzliche Seitengeneratoren, Sitzungsunterstützung und Webseitenmodule. Um eine WebSnap-Web-Server-Anwendung zu erstellen, wählen Sie *Datei / Neu / Weitere*, öffnen die Registerkarte *WebSnap* und doppelklicken auf das Symbol *Web-Server-Anwendung*. WebSnap-Komponenten können Sie aus der Registerkarte *WebSnap* der Komponentenpalette einfügen.

Auf das Datenmodul für WebSnap-Anwendungen können Sie auch zugreifen, indem Sie *Ansicht / Symbolleisten / Internet* wählen und auf das Symbol *Neue WebSnap-Anwendung* klicken



Sie können eine Anwendung erstellen, die auf unterschiedlichen Web-Server-Anwendungen läuft, einschließlich eines Test-Servers, der Sie bei der Fehlersuche in Ihrer Web-Server-Anwendung unterstützt

Sie können auswählen, ob Sie ein Datenmodul erstellen wollen oder ein Seitenmodul, das Ihre HTML-Seite während der Arbeit anzeigt.

Weitere Informationen

Suchen Sie im Hilfeindex nach dem Begriff »Web-Anwendungen«.

Datenbankanwendungen

Um die Entwicklung von Datenbankanwendungen zu vereinfachen, stellt Delphi eine Reihe von Datenbankanwendungen zur Verfügung.

Um eine Datenbank-Anwendung zu erstellen, entwerfen Sie zunächst mit Hilfe der Komponenten der Registerkarte *Datensteuerung* auf einem Formular die Oberfläche. Dann fügen Sie auf der Registerkarte *Datenzugriff* eine Datenquelle zu einem Datenmodul hinzu. Und um schließlich verschiedene Datenbank-Server zu verbinden, fügen Sie eine Datenmenge und eine Datenverbindungskomponente ins Datenmodul ein, und zwar von den vorherigen oder entsprechenden Registerkarten der folgenden Connectivity-Tools:

- dbExpress ist eine Sammlung von Datenbank-Treibern für plattformübergreifende Anwendungen, die einen schnellen Zugriff auf SQL-Datenbank-Server bereitstellen, wie z. B. DB2, InterBase, MySQL oder Oracle. Mit einem dbExpress-Treiber können Sie mit unidirektionalen Datenmengen auf Datenbanken zugreifen.
- Die Borland Database Engine (BDE) ist eine Sammlung von Treibern für viele gängige Datenbank-Formate, einschließlich dBASE, Paradox, FoxPro, Microsoft Access und alle ODBC-Datenquellen. Die mit einigen Delphi-Versionen ausgelieferten SQL-Links-Treiber unterstützen Server wie Oracle, Sybase, Informix, DB2, SQL Server und InterBase.

- ActiveX Data Objects (ADO) ist eine von Microsoft entwickelte High-Level-Schnittstelle zu sämtlichen Datenquellen, einschließlich relationalen und nichtrelationalen Datenbanken, E-Mail- und Dateisystemen, Text und Grafiken sowie zu individuellen Business-Objekten.
- InterBase Express (IBX) basiert auf der Delphi-Komponentenarchitektur für individuell angepaßten Datenzugriff. IBX-Anwendungen gewähren Zugriff auf fortgeschrittene InterBase-Features und bieten die derzeit leistungsfähigste Komponentenschnittstelle für InterBase ab Version 5.5 an. IBX ist kompatibel mit der Bibliothek der datensensitiven Steuerelemente von Delphi.

Manche der Datenbank-Connectivity-Tools sind nur in bestimmten Delphi-Version verfügbar.

Weitere Informationen

Suchen Sie im Hilfeindex nach dem Begriff »Datenbank-Anwendungen«.

BDE-Verwaltung

Die BDE-Verwaltung (BDEAdmin.exe) dient zur Konfiguration von BDE-Treibern und zur Einrichtung der Aliase, die von datensensitiven VCL-Steuerelementen zur Verknüpfung mit Datenbanken benutzt werden.

Weitere Informationen

Wählen Sie im Windows-Startmenü *Programme / Borland Delphi 6 / BDE-Verwaltung*. Wählen Sie anschließend die Menüoption *Hilfe / Inhalt*.

SQL-Explorer (Datenbank-Explorer)

Mit dem SQL-Explorer (DBExplor.exe) lassen sich Datenbanken durchsuchen und bearbeiten. Er kann zur Erstellung von Datenbank-Aliasen verwendet werden, zur Anzeige von Schema-Daten, zur Ausführung von SQL-Abfragen und zur Pflege von Daten-Dictionaries und Attributmengen.

Weitere Informationen

Wählen Sie im Delphi-Hauptmenü *Datenbank / Explorer*. Wählen Sie dann *Hilfe / Inhalt*. Oder suchen Sie im Hilfeindex nach dem Begriff »Datenbank-Explorer«.

Datenbank-Desktop

Mit dem Datenbank-Desktop (DBD32.exe) können Sie Paradox- und dBASE-Tabellen in verschiedenen Formaten erstellen, anzeigen und bearbeiten.

Weitere Informationen

Wählen Sie im Windows-Startmenü *Programme / Borland Delphi 6 / Datenbank-Oberfläche*. Wählen Sie anschließend *Hilfe / Inhalt*.

Daten-Dictionary

Wenn Sie die BDE verwenden, stellt das Daten-Dictionary, unabhängig von Ihren Anwendungen, einen individuell anpaßbaren Speicherbereich zur Verfügung, in dem Sie erweiterte Feldattribute zur Beschreibung des Inhalts und des Erscheinungsbildes von Daten erstellen können. Um einen gemeinsamen Zugriff zu ermöglichen, kann das Daten-Dictionary auf einem Remote-Server eingerichtet werden.

Weitere Informationen

Wählen Sie *Hilfe / Delphi-Tools* und suchen Sie nach »Daten-Dictionary«.

Selbstdefinierte Komponenten

Die mit Delphi ausgelieferten Komponente sind in der Komponentenpalette vorinstalliert und bieten ein Funktionalitätsspektrum, das für die Mehrzahl Ihrer Entwicklungsansprüche ausreichen dürfte. Und doch werden Sie zuweilen das Bedürfnis haben, spezielle Probleme zu lösen oder bestimmte Verhaltensweisen zu kapseln, die eine Definition eigener Komponenten erfordern. Selbstdefinierte Komponenten machen den Code wiederverwendbar und fördern die Konsistenz zwischen verschiedenen Plattformen.

Sie können entweder Komponenten von Drittanbietern beziehen oder eigene definieren. Um eine neue Komponente zu erstellen, wählen Sie *Komponente / Neue Komponente*, wodurch der Experte für neue Komponenten geöffnet wird. Wie Sie Komponenten von Drittanbietern installieren, erfahren Sie im Abschnitt »Komponenten-Packages installieren« auf Seite 5-7.

Weitere Informationen

Lesen Sie Teil V, »Benutzerdefinierte Komponenten erzeugen«, im *Entwicklerhandbuch* sowie die *Object Pascal Sprachreferenz*, oder suchen Sie im Hilfeindex nach dem Begriff »Komponenten erstellen«.

DLLs

DLLs (Dynamic-Link Libraries, dynamische Link-Bibliotheken) sind compilierte Module mit Routinen, die von Anwendungen und von anderen DLLs aufgerufen werden können. Da eine DLL gemeinsam nutzbaren Code bzw. gemeinsam nutzbare Ressourcen enthält, wird sie üblicherweise auch von mehr als einer Anwendung verwendet. Wählen Sie *Datei / Neu / Weitere* und doppelklicken Sie auf das Symbol *DLL-Experte*, um eine DLL-Vorlage als Ausgangspunkt zu verwenden.

Weitere Informationen

Suchen Sie im Hilfeindex nach dem Begriff »DLLs«.

COM und ActiveX

Delphi unterstützt den von Microsoft entwickelten *COM-Standard* (Component Object Model) und enthält Experten für die Erstellung von ActiveX-Steuerelementen. Wählen Sie *Datei* / *Neu* / *Weitere*, und öffnen Sie die Registerkarte *ActiveX*, in der Ihnen die entsprechenden Experten zur Verfügung stehen. Auf der Registerkarte *ActiveX* der Komponentenpalette sind einige Beispiel-ActiveX-Steuerelemente installiert. Auf der Registerkarte *Server* der Komponentenpalette stehen zahlreiche COM-Server-Komponenten zur Verfügung. Sie können diese Komponente so verwenden als wären es normale VCL-Komponenten. So können Sie beispielsweise eine der Microsoft-Word-Komponenten in ein Formular einfügen, um in eine Anwendungsschnittstelle eine Instanz von Microsoft Word zu integrieren.

Weitere Informationen

Suchen Sie im Hilfeindex nach den Begriffen »COM« und »ActiveX«.

Typbibliotheken

Typbibliotheken sind Dateien mit Informationen über Datentypen, Schnittstellen, Elementfunktionen und Objektklassen eines ActiveX-Elements oder ActiveX-Servers. Durch Einbindung einer Typbibliothek in eine COM-Anwendung oder eine ActiveX-Bibliothek stellen Sie anderen Anwendungen oder Programmier-Tools Informationen über diese Objekt zur Verfügung. Mit dem Typbibliotheks-Editor von Delphi können Sie Typbibliotheken erstellen und bearbeiten.

Weitere Informationen

Suchen Sie im Hilfeindex nach dem Begriff »Typbibliotheken«.

Kapitel

4

Die erste Anwendung: ein Texteditor

Dieses Kapitel ist ein Leitfaden für die Erstellung einer ersten Anwendung. Sie lernen von Grund auf, wie Sie in Delphi einen Texteditor programmieren, mit allem was dazugehört: Menüs, eine Symbolleiste und eine Statusleiste.

Hinweis: Die Anleitung gilt für alle Delphi-Versionen, allerdings nur auf Windows-Plattformen.

Eine neue Anwendung beginnen

Bevor Sie eine neue Anwendung beginnen, erstellen Sie ein Verzeichnis für die Quelltextdateien:

1. Erstellen Sie im Verzeichnis PROJECTS des Delphi-Hauptverzeichnisses das Unterverzeichnis TEXTEDITOR.
2. Öffnen Sie ein neues Projekt.

   Jede Anwendung entspricht in Delphi einem *Projekt.* Wenn Sie das Programm starten, wird standardmäßig ein leeres Projekt geöffnet. Ist bereits ein anderes Projekt geöffnet, können Sie mit *Datei / Neu / Anwendung* ein neues Projekt anlegen.

   Für jedes neue Projekt werden automatisch folgende Dateien erstellt:

   - PROJECT1.DPR: eine Quelltextdatei, die mit dem Projekt verbunden ist. Sie wird als *Projektdatei* bezeichnet.
   - UNIT1.PAS: eine Quelltextdatei, die mit dem Hauptformular des Projekts verbunden ist. Diese Datei wird als *Unit-Datei* bezeichnet.
   - UNIT1.DFM: eine Ressourcen-Datei, in der Informationen über das Hauptformular des Projekts gespeichert werden. Sie wird als *Formulardatei* bezeichnet.

   Jedes Formular besitzt seine eigenen Unit- (UNIT1.PAS) und Formulardateien (UNIT1.DFM). Wenn Sie ein zweites Formular erstellen, werden automatisch eine zweite Unit- (UNIT2.PAS) und eine zweite Formulardatei (UNIT2.DFM) erstellt.

3 Wählen Sie *Datei / Alles speichern*.

Wenn das Dialogfenster *Speichern* erscheint, führen Sie folgende Schritte aus:

- Wechseln Sie in das Verzeichnis TEXTEDITOR.
- Speichern Sie die erste Unit unter dem vorgegebenen Namen (UNIT1.PAS).
- Speichern Sie das Projekt unter dem Namen TEXTEDITOR.DPR. (Die EXE-Datei wird denselben Namen erhalten wie das Projekt, nur eben mit der Erweiterung EXE.)

Später können Sie dann jederzeit den Befehl *Datei / Alles speichern* aufrufen, um Ihre Arbeit erneut zu speichern.

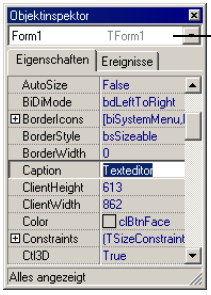
Sie werden jetzt vielleicht feststellen, daß beim Speichern neben den obengenannten Dateien noch weitere Dateien im Projektverzeichnis erstellt wurden, und zwar die Dateien TEXTEDITOR.DOF, die Delphi-Optionendatei, TEXTEDITOR.CFG, die Konfigurationsdatei, sowie TEXTEDITOR.RES, die Windows-Ressourcendatei. Machen Sie sich darüber keine Gedanken. Sie sollten aber diese zusätzlichen Dateien auf keinen Fall löschen.

Eigenschaftswerte festlegen

Wenn Sie in Delphi ein neues Projekt öffnen, wird standardmäßig das Hauptformular mit dem vorgegebenen Namen *Form1* angezeigt. Die Benutzeroberfläche und andere Bestandteile Ihrer Anwendung erstellen Sie, indem Sie Komponenten in dieses Formular einsetzen.

Neben dem Formular befindet sich der Objektinspektor, mit dessen Hilfe Sie die Eigenschaftswerte für das Formular und die darin plazierten Komponenten festlegen können. Die Wartung des Quelltexts können Sie dabei Delphi überlassen. Die auf diese Art festgelegten Eigenschaftswerte werden als *Einstellungen zur Entwurfszeit* bezeichnet.

1 Suchen Sie im Objektinspektor die Formulareigenschaft *Caption* und ersetzen Sie den Standardtitel `Form1` durch `Texteditor`. Beachten Sie, daß beim Eingeben auch der Titel in der Überschrift des Formulars geändert wird.



In der Dropdown-Liste am oberen Rand des Objektinspektors wird das aktuelle Objekt angezeigt. In diesem Fall ist dies *Form1,* das den Typ *TForm1* besitzt

Wenn eine Komponente ausgewählt ist, werden im Objektinspektor ihre Eigenschaften angezeigt

2 Führen Sie das Formular nun aus, indem Sie *F9* drücken, obwohl es noch keine Komponenten besitzt.

Ohne Komponenten ist das Aussehen der Laufzeitansicht ähnlich wie das der Entwurfszeitansicht, einschließlich der Schaltflächen *Minimieren, Maximieren* und *Schließen.*

3 Um die Entwurfszeitansicht von Form1 wieder zu aktivieren, haben Sie folgende Möglichkeiten:

- Klicken Sie rechts oben in der Titelleiste Ihrer Anwendung (der Laufzeitansicht des Formulars) auf das *X*.
- Klicken Sie links in der Titelleiste auf das Anwendungssymbol, und wählen Sie *Schließen.*
- Wählen Sie *Ansicht / Formulare,* wählen Sie *Form1* aus, und klicken Sie auf *OK.*
- Wählen Sie *Ausführen / Programm zurücksetzen.*

Komponenten ins Formular einsetzen

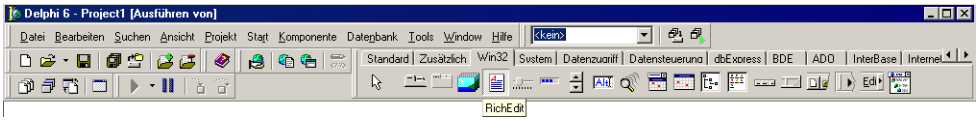
Bevor Sie damit beginnen, Komponenten ins Formular einzufügen, sollten Sie sich überlegen, wie Sie am besten die UI (User Interface, Benutzeroberfläche) Ihrer Anwendung erstellen. Die UI ist der Bereich einer Anwendung, der dem Anwender die Interaktion gestattet, und sollte daher bedienerfreundlich gestaltet sein.

Delphi umfaßt viele Komponenten, die Teile einer Anwendung repräsentieren. So gibt es auf der Komponentenpalette zahlreiche Komponenten, die die Programmierung von Menüs, Symbolleisten, Dialogfenstern und vieler weiterer visueller und nichtvisueller Elemente ganz einfach machen.

Der Texteditor wird einen Bearbeitungsbereich benötigen, eine Statusleiste zur Anzeige von Informationen wie dem Namen der gerade bearbeiteten Datei, einige Menüs und vielleicht eine Symbolleiste mit Schaltflächen, die einen schnellen Zugriff auf wichtige Befehle ermöglichen. Das Schöne an der Oberflächenentwicklung mit Delphi ist die Möglichkeit, mit verschiedenen Komponenten experimentieren und das Resultat unmittelbar überprüfen zu können. Auf diese Weise können Sie schnell den Prototyp einer Programmoberfläche erstellen.

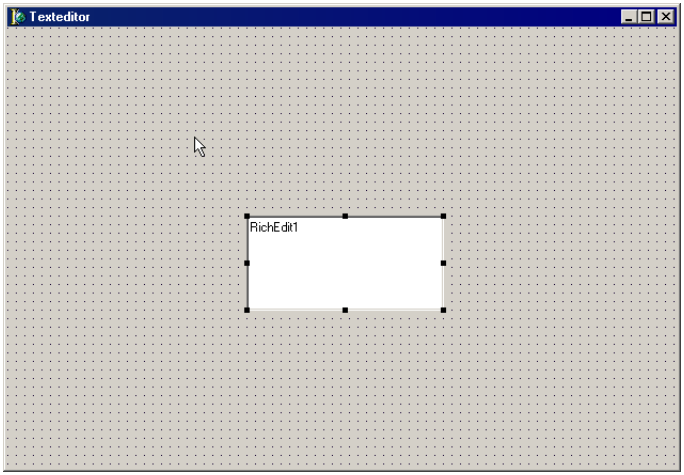
Für den Texteditor fügen Sie zunächst eine *RichEdit*- und eine *StatusBar*-Komponente in das Formular ein:

1 Um einen Textbereich zu erstellen fügen Sie eine *RichEdit*–Komponente ein. Die *RichEdit*-Komponente finden Sie auf der Registerkarte *Win32* der Komponentenpalette. Sie suchen sie, indem Sie den Cursor kurz auf die einzelnen Symbole der Palette setzen, so daß ein Kurzhinweis mit dem Namen der jeweiligen Komponente eingeblendet wird.



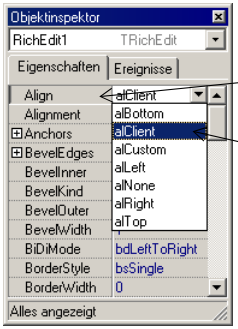
Wenn Sie die *RichEdit*-Komponente gefunden haben, haben Sie folgende Möglichkeiten:

- Um die Komponente an einer bestimmten Stelle zu plazieren, wählen Sie die Komponente auf der Palette aus und klicken dann im Formular auf diese Stelle.
- Um die Komponente in der Mitte des Formulars zu plazieren, doppelklicken Sie auf die Komponente.



Jede Delphi-Komponente stellt eine *Klasse* dar; wenn Sie daher eine Komponente auf einem Formular plazieren, wird eine *Instancz* dieser Klasse erstellt. Sobald sich die Komponente im Formular befindet, erzeugt Delphi den erforderlichen Code, der beim Ausführen der Anwendung eine Instanz des Objekts konstruiert.

2 Solange die *RichEdit*-Komponente noch im Objektinspektor ausgewählt ist, öffnen Sie nun die Dropdown-Liste der Eigenschaft *Align* und setzen sie auf `alClient`.



Im Formular muß die Komponente *RichEdit1* ausgewählt sein

Suchen Sie im Objektinspektor die Eigenschaft *Align* Klicken Sie auf den Abwärtspfeil, um die Dropdown-Liste einer Eigenschaft zu öffnen

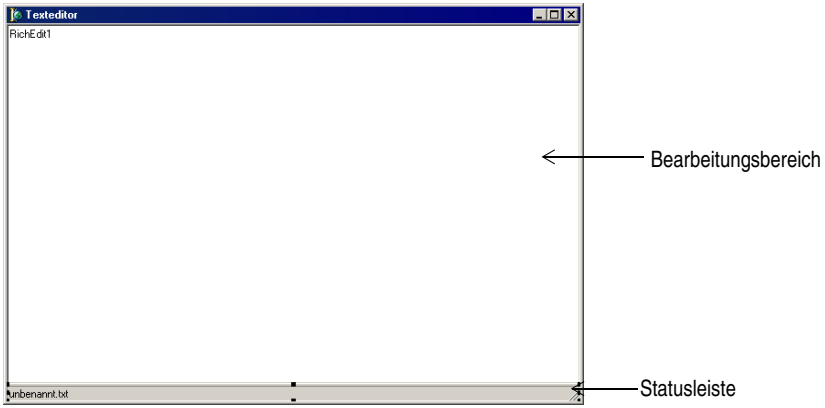
Wählen Sie den Wert `alClient` aus

Die *RichEdit*-Komponente füllt nun das gesamte Formular aus, so daß Sie einen großen Textbearbeitungsbereich haben. Durch die Verwendung des Werts `alClient` für die Eigenschaft *Align* paßt sich die Größe des *RichEdit*-Steuerelements jeweils der angezeigten Fenster- bzw. Formulargröße an.

3 Doppelklicken Sie nun auf der Registerkarte *Win32* der Komponente auf die Komponente *StatusBar*. Dadurch wird das Formular um eine Statusleiste ergänzt.

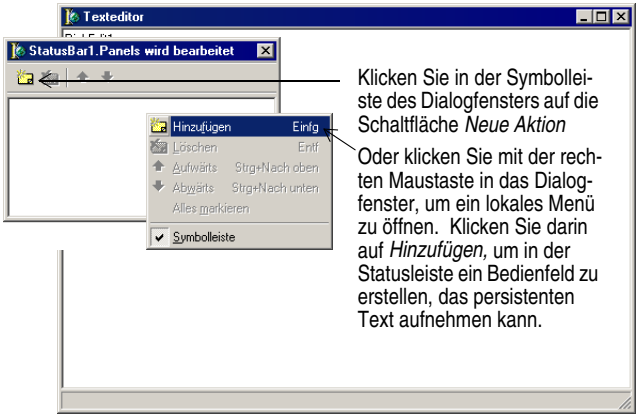
4 Um auf der Statusleiste ein Bedienfeld zu erstellen, das Pfad und Name der in Ihrem Texteditor bearbeiteten Datei anzeigt, führen Sie folgende Schritte aus:

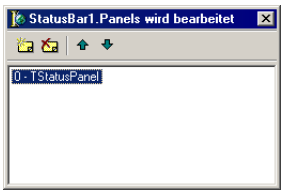
- Die Statusleiste muß aktiviert sein.
- Tragen Sie neben der Eigenschaft *SimpleText* den Wert `unbenannt.txt` ein. Wenn Sie später im Texteditor arbeiten und die aktuelle Datei noch nicht gespeichert haben, so erhält diese den vorläufigen Namen "unbenannt.txt".



- Klicken Sie neben der Eigenschaft *Panels* auf die Schaltfläche mit den drei Punkten zum wert `(TStatusPanel)`. Es erscheint das Dialogfenster *StatusBar1.Panels wird bearbeitet.*
- Um ein Bedienfeld in die Statusleiste einzusetzen, klicken Sie in der Symbolleiste dieses Dialogfensters auf die Schaltfläche *Neue Aktion.*

Tip: Das Dialogfenster *StatusBar1.Panels wird bearbeitet* läßt sich auch öffnen, indem Sie im Formular auf die Statusleiste doppelklicken.





Die Eigenschaft *Panels* ist ein eindimensionales Array, d. h. jedes erstellte Bedienfeld ist über einen eindeutigen Indexwert zugänglich. Standardmäßig besitzt das erste Bedienfeld den Wert 0.

Jedesmal, wenn Sie auf *Hinzufügen* klicken, wird ein weiteres Bedienfeld in die Statusleiste eingesetzt.

5 Schließen Sie das Dialogfenster *StatusBar1.Panels wird bearbeitet*, indem Sie auf *X* klicken.

Der Hauptbearbeitungsbereich der Benutzeroberfläche für den Texteditor ist jetzt eingerichtet.

Menü- und Symbolleistenunterstützung hinzufügen

Damit eine Anwendung überhaupt aktionsfähig ist, benötigt sie ein Menü, Befehle und, aus Gründen der leichteren Bedienbarkeit, eine Symbolleiste. Es ist zwar auch möglich, die Befehle gesondert zu programmieren, doch verfügt Delphi über einen *Aktions-Manager,* der Sie bei einer zentralen Verwaltung des Quelltexts unterstützt, und über eine *Bildliste,* in der die Bilder zentral zusammengeführt werden, die Sie den Befehlen auf Ihren Menüs und der Symbolleiste zuweisen können.

In der Regel werden die Aktionen, die mit den Menübefehlen verbunden werden, nach dem Menü auf der obersten Ebene und dem Befehlsnamen benannt. So bezeichnet die Aktion *DateiBeenden* etwa den Befehl *Beenden* im Menü *Datei.*

Es folgt eine Liste der Aktionen, die der Texteditor der Beispielanwendung benötigt.

Tabelle 4.1 Die geplanten Texteditor-Befehle

Menü	Befehl	In der Symbolleiste?	Beschreibung
Datei	Neu	Ja	Erstellt eine neue Datei.
Datei	Öffnen	Ja	Öffnet eine vorhandene Datei für die Bearbeitung.
Datei	Speichern	Ja	Speichert die aktuelle Datei auf einem Datenträger.

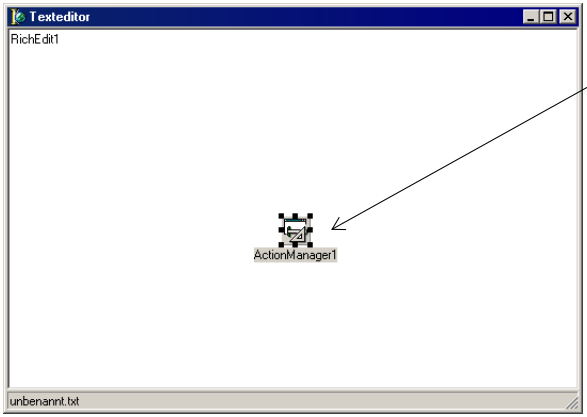
Tabelle 4.1 Die geplanten Texteditor-Befehle (Fortsetzung)

Menü	Befehl	In der Symbolleiste?	Beschreibung
Datei	Speichern unter	Nein	Speichert eine Datei unter einem neuen Namen (erlaubt auch die Speicherung einer neuen Datei unter einem bestimmten Namen).
Datei	Beenden	Ja	Beendet das Editor-Programm.
Bearbeiten	Ausschneiden	Ja	Löscht Text und speichert ihn gleichzeitig in die Zwischenablage.
Bearbeiten	Kopieren	Ja	Kopiert Text, indem er ihn in die Zwischenablage gespeichert wird.
Bearbeiten	Einfügen	Ja	Fügt den in der Zwischenablage befindlichen Text ein.
Hilfe	Inhalt	Nein	Öffnet das Hilfefenster *Inhalt*, das ein Inhaltsverzeichnis über die Hilfethemen darstellt.
Hilfe	Index	Nein	Öffnet das Hilfefenster *Index*.
Hilfe	Info	Nein	Öffnet ein Fenster mit Informationen über die Anwendung.

Um den Quelltext und die Bilder in einem Aktions-Manager zentral zu verwalten, müssen Sie den Aktions-Manager in Ihr Projekt einfügen:

1 Doppelklicken Sie auf der Registerkarte *Zusätzlich* der Komponentenpalette auf die Komponente *ActionManager*, so daß sie ins Formular eingesetzt wird. Da es sich hier um eine nichtvisuelle Komponente handelt, können Sie sie an beliebiger Stelle im Formular plazieren.

2 Damit im Formular die Titel nichtvisueller Komponenten angezeigt werden, wählen Sie *Tools / Umgebungsoptionen*, öffnen die Registerkarte *Designer*, aktivieren die Option *Komponenten-Titel zeigen*, und klicken auf *OK*.



Damit im Formular die Titel der Komponenten, die Sie im Formular plaziert haben, angezeigt werden, wählen Sie *Tools / Umgebungsoptionen / Designer* und aktivieren die Option *Komponenten-Titel zeigen*

Da der *ActionManager* eine nichtvisuelle Komponente ist, ist sie nicht sichtbar, wenn die Anwendung ausgeführt wird.

Aktionen in den Aktions-Manager einfügen

Zunächst werden Sie die Aktionen in den Aktions-Manager einfügen und ihre Eigenschaften festlegen. In der Regel werden die Aktionen, die mit den Menübefehlen verbunden werden, nach dem Menü auf der obersten Ebene und dem Befehlsnamen benannt. So bezeichnet die Aktion *DateiBeenden* etwa den Befehl *Beenden* im Menü *Datei.*

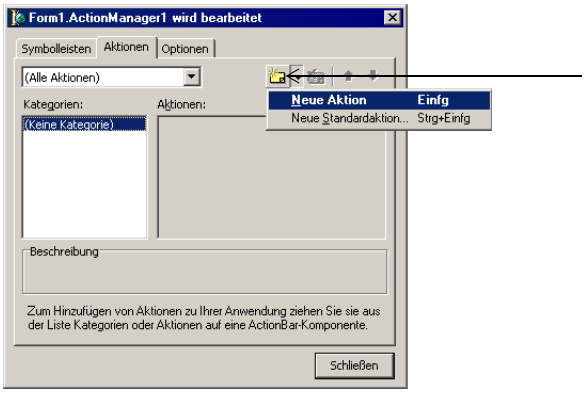
Sie werden Aktionen einfügen, für die Sie alle Eigenschaft festlegen, wie auch Standardaktionen, deren Eigenschaften automatisch eingestellt werden.

1 Doppelklicken Sie auf die Komponente *ActionManager.*

Das Dialogfenster *Form1.ActionManager1 wird bearbeitet* (der Aktions-Manager) erscheint.

2 Öffnen Sie das Registerkarte *Aktionen*. Öffnen Sie die Dropdown-Liste neben der Schaltfläche *Neue Aktion,* und wählen Sie den Eintrag *Neue Aktion* aus.

Tip: Sie können auch mit der rechten Maustaste auf den Aktions-Manager-Editor klicken und *Neue Aktion* wählen.



Öffnen Sie die Dropdown-Liste neben der Schaltfläche *Neue Aktion,* um neue Aktionen für den Aktions-Manager zu erstellen.

Ist die Schaltfläche *Löschen* aktiviert, so können Sie vorhandene Aktionen aus der Aktionsliste entfernen.

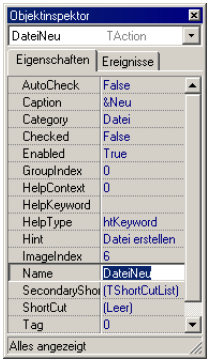
3 Wählen Sie in der Aktionsliste *Keine Kategorie* aus, und klicken Sie auf *Action1*. Legen Sie im Objektinspektor die folgenden Eigenschaften fest:

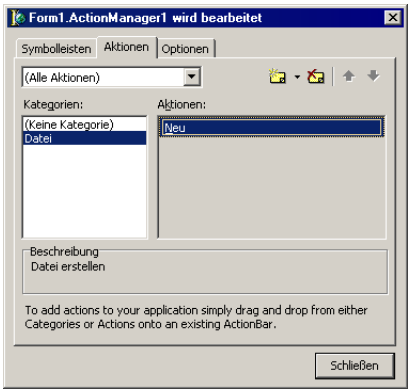
- Tragen Sie als *Caption* den Wert `&Neu` ein. (Die Eingabe des Zeichens & vor dem Titel bewirkt, daß der nachfolgende Buchstabe später unterstrichen ist und als Tastenkürzel dient.)
- Geben Sie als *Category* `Datei` ein (dadurch werden alle Dateibefehle zusammen angeordnet).
- Als *Hint* tragen Sie `Datei erstellen ein` (dieser Text erscheint später als Kurzhinweis).
- Hinter *ImageIndex* tragen Sie `6` ein (dadurch wird Bild Nr. 6 in der Bildliste mit dieser Aktion verknüpft).

- Als *Name* geben Sie `DateiNeu` ein (stellvertretend für den Befehl *Datei / Neu*) und drücken *Eingabe*, um die Änderung zu speichern.

Aktivieren Sie im Aktions-Manager-Editor *Action1* und ändern Sie die Eigenschaften dieser Aktion im Objektinspektor

Caption ist der Titel der Aktion, *Category* die Art, *Hint* ist ein Kurzhinweis, *ImageIndex* erlaubt den Verweis auf ein Bild in der Bildliste und *Name* ist der Name der Aktion im Quelltext.





4 Öffnen Sie die Dropdown-Liste neben der Schaltfläche *Neue Aktion*, und wählen Sie den Eintrag *Neue Aktion* aus.

5 Wählen Sie *No Category* aus, und klicken Sie auf *Action1*. Legen Sie im Objektinspektor die folgenden Eigenschaften fest:

- Geben Sie hinter *Caption* den Titel `&Speichern ein`.
- Wählen Sie aus der Dropdown-Liste der *Category* den Eintrag `Datei` aus.
- Neben *Hint* geben Sie den Kurzhinweis `Datei speichern` ein.
- Als *ImageIndex* geben Sie `8 ein`.
- Tragen Sie als *Name* `DateiSpeichern` ein (für den Befehl *Datei / Speichern*).

6 Öffnen Sie die Dropdown-Liste neben der Schaltfläche *Neue Aktion*, und wählen Sie den Eintrag *Neue Aktion* aus.

7 Wählen Sie *No Category* aus, und klicken Sie auf *Action1*. Legen Sie im Objektinspektor die folgenden Eigenschaften fest:

- Geben Sie hinter *Caption* den Titel `&Index ein`.
- Tragen Sie als *Category* `Hilfe` ein.
- Tragen Sie als *Name* `HilfeIndex` ein (für den Befehl *Hilfe / Index*).

8 Öffnen Sie die Dropdown-Liste neben der Schaltfläche *Neue Aktion*, und wählen Sie den Eintrag *Neue Aktion* aus.

9 Wählen Sie neben *(No Category) Action1* aus. Legen Sie im Objektinspektor die folgenden Eigenschaften fest:

- Geben Sie hinter *Caption* den Titel `&Info ein`.
- Als *Category* muß `Hilfe` ausgewählt sein.
- Tragen Sie als *Name* `HilfeInfo` ein (für den Befehl *Hilfe / Info*).

10 Lassen Sie den Aktions-Manager-Editor aktiviert.

Standardaktionen in den Aktions-Manager einfügen

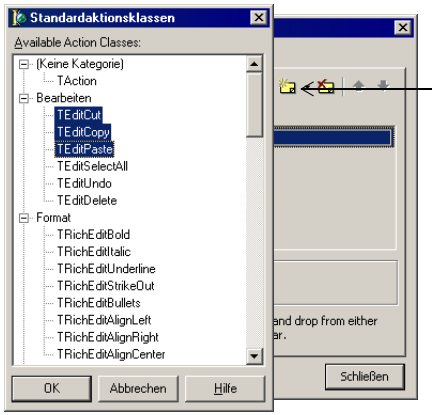
Als Nächstes werden Sie die Standardaktionen *(Öffnen, Speichern unter, Beenden, Ausschneiden, Kopieren, Einfügen und Hilfe / Inhalt)* in den Aktions-Manager einfügen.

1 Der Aktions-Manager-Editor sollte immer noch angezeigt werden. Doppelklicken Sie auf die Komponente *ActionManager*, um sie zu öffnen.

2 Öffnen Sie die Dropdown-Liste neben der Schaltfläche *Neue Aktion*, und wählen Sie den Eintrag *Neue Standardaktion* aus.

Das Dialogfenster *Standardaktionsklassen* erscheint.

3 Zeigen Sie in diesem Dialogfenster die Kategorie *Bearbeiten* an und wählen Sie *TEditCut, TEditCopy* und *TEditPaste* aus. Klicken Sie auf *OK*, und diese Aktionen werden in eine neue Bearbeiten-Kategorie in der Kategorienliste des Dialogfensters *Form1.ActionManager1 wird bearbeitet* aufgenommen.



Klicken Sie auf die Schaltfläche *Neue Aktion* und wählen Sie *Neue Standardaktion.*

Es werden die neuen Standardaktionen angezeigt. Um eine auszuwählen doppelklicken auf die Aktion.

4 Öffnen Sie die Dropdown-Liste neben der Schaltfläche *Neue Aktion*, und wählen Sie den Eintrag *Neue Standardaktion* aus.

5 Zeigen Sie die Kategorie *Datei* an und wählen Sie die Aktionen *TFileOpen, TFileSaveAs* und *TFileExit* aus. Klicken Sie auf *OK*, um diese Aktionen in die Kategorie *Datei* aufzunehmen.

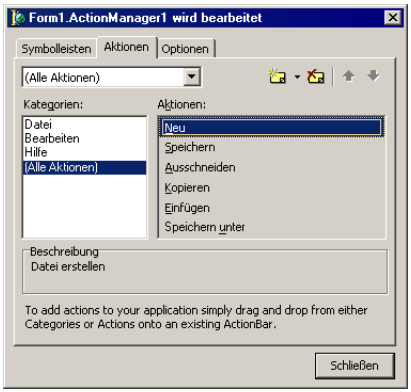
6 Öffnen Sie die Dropdown-Liste neben der Schaltfläche *Neue Aktion*, und wählen Sie den Eintrag *Neue Standardaktion* aus.

7 Zeigen Sie im Dialogfenster *Standardaktionsklassen* die Kategorie *Hilfe* an und wählen Sie *THelpContents* aus. Klicken Sie auf *OK*, um diese Aktion in die Kategorie *Hilfe* aufzunehmen.

Hinweis: Wenn Sie einen selbst definierten Befehl *Hilfe / Inhalt* hinzufügen, wird eine Hilfedatei mit der Registerkarte *Inhalt* angezeigt. Der Standardbefehl *Hilfe / Inhalt* öffnet die letzte Registerkarte, die angezeigt worden ist, also *Inhalt, Index* oder *Suchen.*

Jetzt haben Sie alle Standardaktionen für die Anwendung hinzugefügt. Für Standardaktionen werden die Eigenschaften automatisch eingestellt, einschließlich des Bildindex.

8 Klicken Sie auf *(Alle Aktionen)*, damit sowohl die selbst definierten als auch die Standardaktionen, die Sie soeben hinzugefügt haben, angezeigt werden.



Per Klick auf *Alle Aktionen* werden zu jeder Kategorie die soeben definierten Aktionen angezeigt.

9 Klicken Sie auf die Schaltfläche *Schließen*, damit der Aktions-Manager-Editor geschlossen wird.

10 Wählen Sie *Datei / Alles speichern*, damit Ihre Änderungen gespeichert werden.

Bilder in die Bildliste einfügen

In diesem Abschnitt werden Sie Bilder in den Aktions-Manager einfügen, damit sie in den Menüs und der Symbolleiste verwendet werden können.

Die Standardaktionen werden mit bereits zugewiesenen Bilder verknüpft aus einer integrierten Bildliste, die mit Delphi ausgeliefert wird. So besitzt die Aktion *Bearbeiten / Ausschneiden* beispielsweise die Bildindexnummer 0. Alle Bilder, die Sie für die Texteditorbefehle verwenden werden, befinden sich in dieser Datei.

Um Bilder in die Bildliste einzufügen, führen Sie folgende Arbeitsschritte aus:

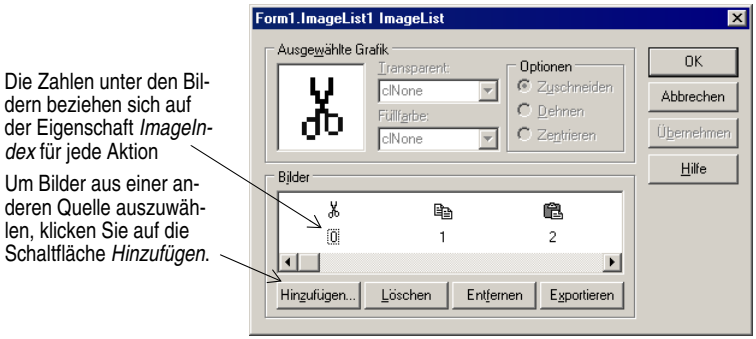
1 Wenn Sie Delphi im Standardverzeichnis installiert haben, öffnen Sie C:\Programme\Borland\Delphi6\Source\Vcl\ActnRes.pas. Das Fenster *StandardActions* wird geöffnet.

2 Wählen Sie die Komponente *ImageList1* aus, kopieren Sie sie, und fügen Sie sie in Ihr Formular ein. Es handelt sich um eine nichtvisuelle Komponente, d. h. es ist unwichtig, wo Sie sie genau einfügen. In den Quelltext-Editor wird die Unit ActnRes.pas eingefügt.

- Um *ImageList1* zu kopieren, klicken Sie mit der rechten Maustaste auf die Komponente und wählen *Bearbeiten / Kopieren*. Klicken Sie mit der rechten Maustaste auf das Formular und wählen Sie *Bearbeiten / Einfügen*.

3 Schließen Sie das Fenster *StandardActions*.

4 Doppelklicken Sie auf *ImageList1*, damit alle Bilder angezeigt werden, die Sie verwenden können.



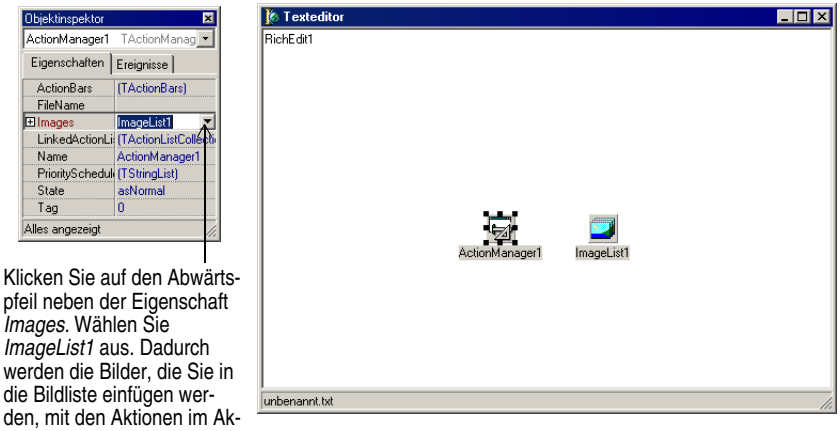
Es folgt eine Liste der Bildindexnummern, die für jeden Befehl verwendet werden:

Befehl	Eigenschaft ImageIndex
Bearbeiten / Ausschneiden	0
Bearbeiten / Kopieren	1
Bearbeiten / Einfügen	2
Datei / Neu	6
Datei / Öffnen	7
Datei / Speichern	8
Datei / Speichern unter	30
Datei / Beenden	43
Hilfe / Inhalt	40

Hinweis: Sie können auch Bilder aus einer völlig anderen Liste hinzufügen. Dazu klicken Sie im Dialogfenster *Form1.ImageList* auf die Schaltfläche *Hinzufügen* und öffnen das vom Projekt bereitgestellte Schaltflächenverzeichnis. Das Standardverzeichnis ist C:\Programme\Gemeinsame Dateien\Borland Shared\Images\Buttons. Für den Befehl *Datei / Öffnen* beispielsweise doppelklicken Sie auf fileopen.bmp. Wenn Sie in einer Meldung gefragt werden, ob Sie die Bitmap-Grafik in zwei einzelne aufspalten wollen, klicken Sie auf *Ja.* Zu jedem Bild gehört eine aktive und eine abgeblendete Version des Bilds. Sie werden hier beide Bilder sehen. Löschen Sie das abgeblendete (zweite) Bild. Stellen Sie dann sicher, daß der Bildindex im Objektinspektor mit der neuen, diesem Bild in der Bildliste zugewiesenen Nummer übereinstimmt.

5 Klicken Sie auf *OK*, um das Dialogfenster *ImageList* zu schließen.

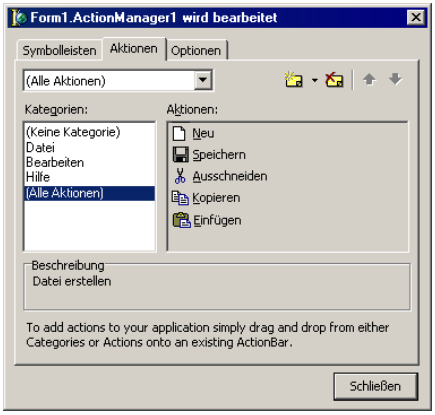
6 Wählen Sie die Komponente *ActionManager* aus, und setzen Sie die Eigenschaft *Images* auf `ImageList1`.



Klicken Sie auf den Abwärtspfeil neben der Eigenschaft *Images*. Wählen Sie *ImageList1* aus. Dadurch werden die Bilder, die Sie in die Bildliste einfügen werden, mit den Aktionen im Aktions-Manager verknüpft.

Da Sie bereits einen Bildindex für alle Ihre Aktionen festgelegt haben, werden die Bilder automatisch der richtigen Aktion zugeordnet. Sie haben acht Bilder mit Ihren Aktionen verknüpft.

7 Um die verknüpften Bilder im Aktions-Manager anzuzeigen, öffnen Sie die Komponente *ActionManager* öffnen die Registerkarte *Aktionen* und klicken auf die Kategorie *Alle Aktionen*.



Wenn Sie jetzt den Aktions-Manager-Editor anzeigen, werden Sie die mit den Aktionen verbundenen Bilder sehen.

8 Wählen Sie *Datei / Alles speichern*, damit Ihre Änderungen gespeichert werden.

9 Lassen Sie den Aktions-Manager-Editor geöffnet.

Jetzt können Sie das Menü und die Symbolleiste hinzufügen.

Ein Menü hinzufügen

In den nächsten beiden Abschnitten werden Sie eine anpaßbare Menüleiste und Symbolleiste, sogenannte *Aktionsbänder* hinzufügen.

Die Hauptmenüleiste umfaßt drei Dropdown-Menüs – *Datei*, *Bearbeiten* und *Hilfe* – mit den jeweiligen Menübefehlen. Mit dem Aktions-Manager-Editor können Sie jede Menükategorie mit ihren Befehlen in einem Schritt auf die Menüleiste ziehen.

1. Doppelklicken Sie auf der Registerkarte *Zusätzlich* der Komponentenpalette auf die Komponente *ActionMainMenuBar*, so daß sie ins Formular eingesetzt wird.

   Dadurch wird oben im Formular eine leere Menüleiste eingesetzt.

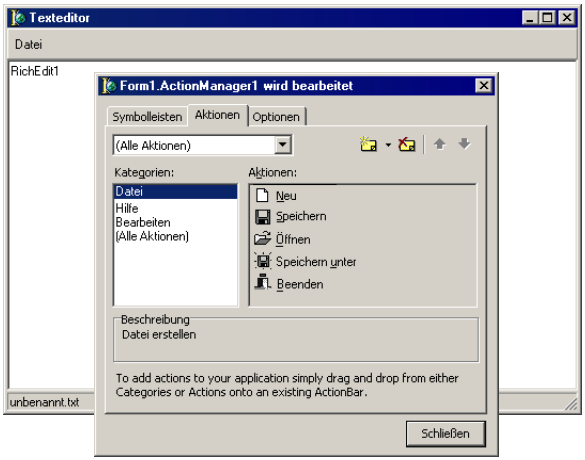
2. Öffnen Sie, falls erforderlich, den Aktions-Manager-Editor, und wählen Sie in der Kategorienliste den Eintrag *Datei* aus. Möglicherweise befinden sich die Menübefehle noch nicht in der gewünschten Reihenfolge, doch können Sie das mit Hilfe der Schaltflächen *Nach oben* und *Nach unten* bzw. mit *Umschalt+*↑ und *Umschalt+*↓ leicht ändern.

   - Markieren Sie die Aktion *Öffnen*, und klicken Sie in der Symbolleiste des Aktions-Manager-Editors auf die Schaltfläche *Nach oben*, so daß die Befehle des Menüs *Datei* in der folgenden Reihenfolge aufgelistet werden: *Neu*, *Öffnen*, *Speichern*, *Speichern unter*, *Beenden*.

3. Ziehen Sie *Datei* auf die Menüleiste. Das Menü *Datei* erscheint mit seinen Menübefehlen auf der Menüleiste.

Tip: Die Menübefehle lassen sich auch noch umarrangieren, nachdem Sie sie auf die Menükategorie der Menüleiste gezogen haben. Klicken Sie dazu versuchsweise in der Menüleiste auf *Datei*, damit die Befehle des Menüs ausgeklappt werden, und ziehen Sie den Befehl *Öffnen* über den Befehl *Neu*, und wieder zurück.

Wenn Sie die Kategorie *Datei* vom Aktions-Manager-Editor auf die Menüleiste ziehen, werden gleichzeitig auch alle Befehle dieses Menüs verlagert .



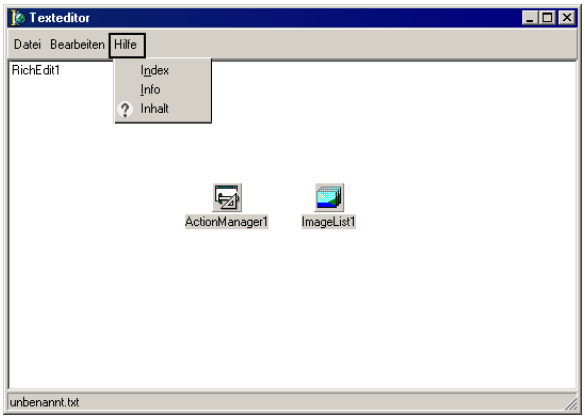
4. Ziehen Sie *Bearbeiten* von der Kategorienliste des Aktions-Manager-Editors auf die Menüleiste, und zwar rechts neben das Menü *Datei*.

5 Ziehen Sie *Hilfe* von der Kategorienliste des Aktions-Manager-Editors auf die Menüleiste, und zwar rechts neben das Menü *Bearbeiten*.

6 Klicken Sie auf das Menü *Hilfe*, damit seine Menübefehle angezeigt werden. Ziehen Sie den Befehl *Inhalt* über den Befehl *Index*.

Sie können die Position von Menübefehlen auf zwei Arten ändern:

Sie markieren eine Aktion im Aktions-Manager-Editor, und klicken dann auf die Schaltfläche *Nach oben* bzw. *Nach unten*. Oder Sie ziehen die Kategorie *Hilfe* zunächst auf die Menüleiste und ziehen dann den Befehl *Inhalt* über den Befehl *Info*.



7 Drücken Sie *Esc* oder klicken Sie erneut auf das Menü *Hilfe*, um es zu schließen.

8 Wählen Sie *Datei / Alles speichern*, damit Ihre Änderungen gespeichert werden.

Nun werden Sie noch eine Symbolleiste hinzufügen, um die Befehle leichter zugänglich zu machen.

Eine Symbolleiste hinzufügen

Da Sie im Aktions-Manager bereits Aktionen eingerichtet haben, können Sie einige dieser Aktionen, die schon in den Menüs verwendet worden sind, in eine Aktionsband-Symbolleiste einsetzen, die am Ende einer Microsoft-Office-2000-Symbolleiste ähneln wird.

1 Doppelklicken Sie auf der Registerkarte *Zusätzlich* der Komponentenpalette auf die Komponente *ActionToolBar*, so daß sie ins Formular eingesetzt wird.

Unter der Menüleiste erscheint eine leere Symbolleiste.

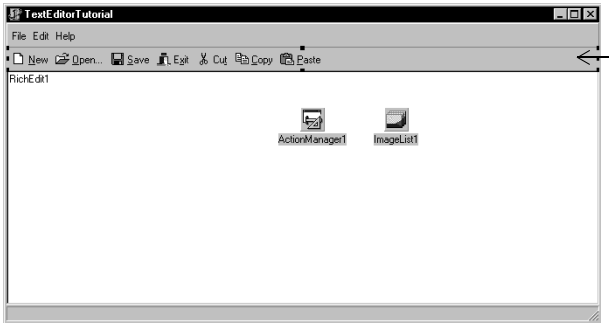
Tip: Sie können eine Aktionsband-Symbolleiste auch hinzufügen, indem Sie den Aktions-Manager-Editor öffnen und auf der Registerkarte *Symbolleisten* auf die Schaltfläche *Neu* klicken.

2 Öffnen Sie, falls erforderlich, den Aktions-Manager-Editor, und wählen Sie in der Kategorienliste den Eintrag *Datei* aus.

3 Markieren Sie in der Aktionsliste die Einträge *Neu*, *Öffnen*, *Speichern* und *Beenden*, und ziehen Sie diese auf die Symbolleiste. Sie erscheinen automatisch als Befehlssymbole mit den jeweils zugewiesenen Bildern.

4 Markieren Sie in der Kategorienliste des Aktions-Manager-Editors den Eintrag *Bearbeiten*.

- Markieren Sie in der Aktionsliste die Einträge *Ausschneiden*, *Kopieren* und *Öffnen*, und ziehen Sie diese auf die Symbolleiste.



Die Komponente *ActionToolBar* wird standardmäßig unter der Menüleiste angeordnet

Durch einfaches Ziehen können Sie die Symbolleiste über die Menüleiste verlagern, und umgekehrt

Sie können die Befehlssymbole auf die Symbolleiste ziehen und zum Entfernen einfach wieder wegziehen.

Sollten Sie versehentlich den falschen Befehl auf die Symbolleiste gezogen haben, entfernen Sie ihn wieder, indem Sie ihn einfach von der Symbolleiste herunterziehen. Oder Sie markieren das Objekt in der Objekthierarchie und drücken die Taste *Entf*. Sie ändern die Reihenfolge der Befehlsymbole, indem Sie sie nach recht oder links ziehen.

5 Wählen Sie *Datei / Alles speichern*, damit Ihre Änderungen gespeichert werden.

6 Drücken Sie *F9*, damit das Projekt compiliert und ausgeführt wird.

Tip: Sie können ein Projekt auch ausführen, indem Sie auf der Debug-Symbolleiste auf die Schaltfläche *Start* klicken oder in der Menüleiste *Start / Start* wählen.

Wenn Sie Ihr Projekt ausführen, öffnet Delphi das Programm in einem Laufzeit-Fenster, das dem von Ihnen entworfenen Anwendungsfenster entspricht. Die Menüs und Symbolleisten funktionieren bereits, obwohl einige der Befehl abgeblendet sind.

Sie können schon einiges machen in Ihrem Texteditor: Zum Beispiel Text in den Textbereich eingeben oder Text im Textbereich markieren und die Befehlssymbole *Ausschneiden*, *Kopieren* und *Einfügen* darauf anwenden. Allerdings gibt es noch etwas Arbeit, um die Befehle zu aktivieren.

7 Um den Entwurfsmodus wieder zu aktivieren, klicken Sie rechts oben im Laufzeit-Fenster auf das *X*.

Den Inhalt des Textbereichs entfernen (optional)

Beim Ausführen Ihres Programms, erschien im Textbereicht der Name *RichEdit1*. Diesen Text können Sie mit Hilfe des String-Listen-Editors entfernen. Wenn Sie diesen Text jetzt nicht entfernen, sollte er im letzten Schritt, bei der Initialisierung des Hauptformulars, entfernt werden.

Um den Inhalt des Textbereichs zu entfernen, führen Sie folgende Schritte aus:

1 Klicken Sie im Hauptformular auf die Komponente *RichEdit1*.

2 Doppelklicken Sie im Objektinspektor neben der Eigenschaft *Lines* auf den Wert *(TStrings).* Der Filter-Editor erscheint.

3 Markieren Sie im Filter-Editor den zu entfernenden Text *(RichEdit1),* und klicken Sie auf *OK.*

4 Speichern Sie Ihre Änderungen, und führen Sie das Programm erneut aus.

Der Textbearbeitungsbereich ist jetzt leer, wenn das Hauptformular angezeigt wird.

Ereignisbehandlungsroutinen schreiben

Bisher haben Sie ihre Anwendung geschrieben ohne eine einzige Zeile Code zu schreiben. Indem Sie beim Entwurf den Objektinspektor verwendet haben, um die Werte der Eigenschaften festzulegen, haben Sie die Vorteile der RAD-Umgebung von Delphi voll genutzt. In diesem Abschnitt werden Sie Prozeduren schreiben, die auf beim Ausführen der Anwendung auf Benutzereingaben reagieren: sogenannte *Ereignisbehandlungsroutinen.* Sie werden die Ereignisbehandlungsroutinen mit den Objekten in den Menüs und auf der Symbolleiste verbinden, so daß bei Auswahl eines Objekts die Anwendung den Code in der Ereignisbehandlungsroutine ausführt.

Ereignisbehandlungsroutinen müssen Sie nur für Aktionen schreiben, die keine Standardaktionen sind. Bei Standardaktionen wie *Datei / Beenden* oder *Bearbeiten / Einfügen* sind die Ereignisse bereits in den Code eingebunden. Doch ist es für manche Standardaktionen, wie etwa *Datei / Speichern unter,* oft vorzuziehen, eine eigene Ereignisbehandlungsroutine zu schreiben und den Befehl abzuwandeln.

Da die Aktionen aller Menüoptionen und Befehlssymbole im Aktions-Manager-Editor zusammengetragen sind, erstellen Sie die Ereignisbehandlungsroutinen am besten von dort aus.

Eine Ereignisbehandlungsroutine für den Befehl Neu

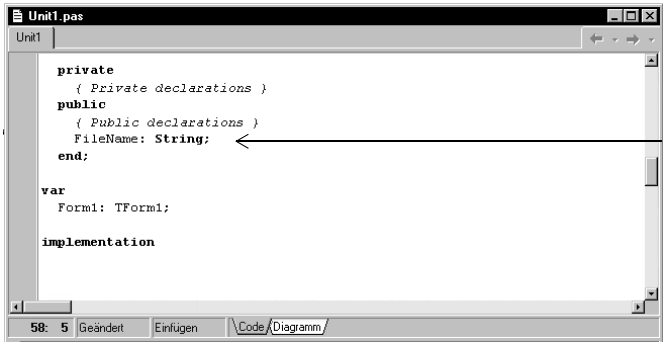
Um eine Ereignisbehandlungsroutine für den Befehl *Neu* zu erstellen, führen Sie folgende Arbeitsschritte aus:

1 Wählen Sie *Ansicht / Units,* und markieren Sie *Unit1,* damit der mit Form1 verbundene Quelltext angezeigt wird.

2 Sie müssen einen Dateinamen deklarieren, der Ereignisbehandlungsroutine verwendet wird, und eine selbst definierte Eigenschaft für den Dateinamen einfügen, um diesen global zugänglich zu machen. Suchen Sie am Anfang der Datei Unit1.pas den Abschnitt mit den Public-Deklarationen für die Klasse *TForm1,* und geben Sie hinter der Zeile `{ Puarations }` den folgenden Text ein:

```
FileName: String;
```

Auf dem Bildschirm sollte jetzt Folgendes zu sehen sein:



3 Drücken Sie *F12,* um das Hauptformular wieder zu aktivieren.

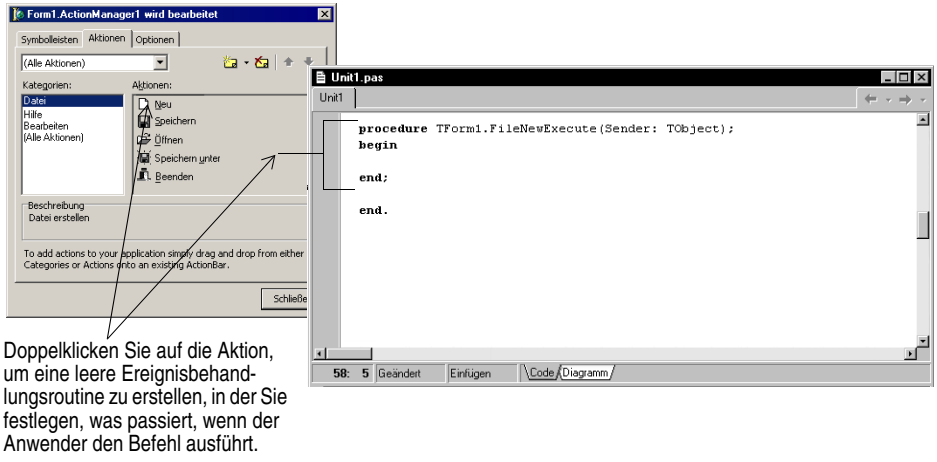
Tip: *F12* fungiert als Schalter zwischen dem Formular und dem zugehörigen Quelltext.

4 Doppelklicken Sie auf den *ActionManager*.

5 Markieren Sie im Aktions-Manager-Editor die Kategorie *Datei*, und doppelklicken Sie dann auf die Aktion *Neu*.

Tip: Sie können auch in der Objekthierarchie auf die Aktion *Datei/DateiNeu* doppelklikken.

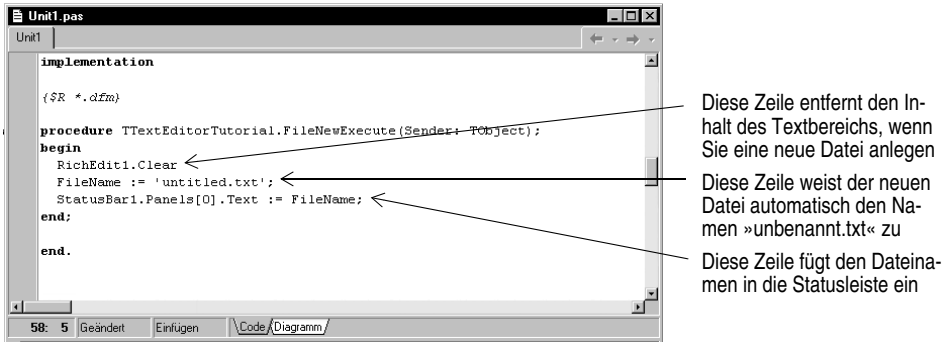
Der Quelltext-Editor wird geöffnet, und der Cursor befindet sich bereits innerhalb der Ereignisbehandlungsroutine.



6 Geben Sie direkt an der Cursorposition (zwischen `begin` und `end`) die folgenden Zeilen ein:

```
RichEdit1.Clear;
FileName := 'unbenannt.txt';
StatusBar1.Panels[0].Text := FileName;
```

Wenn Sie fertig sind, sollte die Ereignisbehandlungsroutine folgenden Inhalt haben:



Speichern Sie Ihre Arbeit, und das war's auch schon, was den Befehl *Datei / Neu* anbelangt.

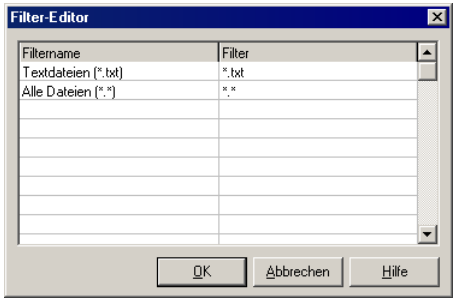
Hinweis: Sie können den Fensterbereich für den Quelltext vergrößern, damit weniger Bildläufe erforderlich sind.

Eine Ereignisbehandlungsroutine für den Befehl Öffnen

Beim Öffnen einer Texteditordatei soll das Standard-Windows-Fenster für das Öffnen von Dateien erscheinen. Den Standardbefehl *Datei / Öffnen* haben Sie schon in den Aktions-Manager-Editor eingefügt. Dieser Befehl beinhaltet das Dialogfenster bereits. Dennoch soll die Ereignisbehandlungsroutine für den Befehl etwas abgeändert werden:

1. Drücken Sie *F12*, damit das Hauptformular angezeigt wird (oder wählen Sie *Ansicht / Formulare*, und wählen Sie *Form1*).
2. Öffnen Sie den Aktions-Manager-Editor. Markieren Sie die Aktion *Datei / Öffnen*.
3. Doppelklicken Sie im Objektinspektor neben der Eigenschaft *Dialog* auf das Pluszeichen, damit die verfügbaren Eigenschaften eingeblendet werden. Standardmäßig vergibt Delphi für dieses Dialogfenster den Namen *FileOpen1.OpenDialog*. Bei Aufruf der *OpenDialog1*-Methode *Execute* wird das Standarddialogfenster für das Öffnen von Dateien aufgerufen.
4. Legen Sie für *FileOpen1.Dialog* die folgenden Eigenschaften fest:
   - Setzen Sie *DefaultExt* auf `txt`.
   - Doppelklicken Sie auf den Textbereich neben *Filter*, um den Filter-Editor anzuzeigen. Tragen Sie in die erste Zeile unter der Spalte *Filter Name* den Text `Textdateien (*.txt)` ein. Geben Sie in die Spalte *Filter* `*.txt` ein. Tragen Sie in die

zweite Zeile unter der Spalte *Filter Name* den Text `Alle Dateien (*.*)` ein und in die Spalte *Filter* den Text `*.*`. Klicken Sie auf *OK*.



Definieren Sie die Filter für die Aktionen *FileOpen1.Dialog* und *FileSaveAs1.Dialog* anhand des Filter-Editors.

- Neben *Title* geben Sie `Datei öffnen` ein. Dieser Text wird in der Titelleiste des Dialogfensters erscheinen.

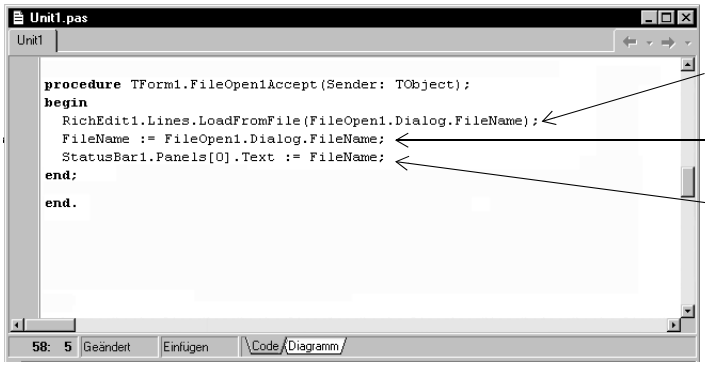
5 Öffnen Sie die Registerkarte *Ereignisse*. Doppelklicken Sie auf das Ereignis *OnAccept*, so daß `FileOpen1Accept` erscheint.

6 Der Quelltext-Editor wird geöffnet, und der Cursor befindet sich bereits innerhalb der Ereignisbehandlungsroutine.

7 Geben Sie direkt an der Cursorposition (zwischen **`begin`** und **`end`**) die folgenden Zeilen ein:

```
RichEdit1.Lines.LoadFromFile(FileOpen1.Dialog.FileName);
FileName := FileOpen1.Dialog.FileName;
StatusBar1.Panels[0].Text := FileName;
```

Wenn Sie fertig sind, sollte die Ereignisbehandlungsroutine *FileOpen* folgenden Inhalt haben:



Das war alles zum Befehl *Datei / Öffnen* und dem zugehörigen Dialogfenster.

Eine Ereignisbehandlungsroutine für den Befehl Speichern

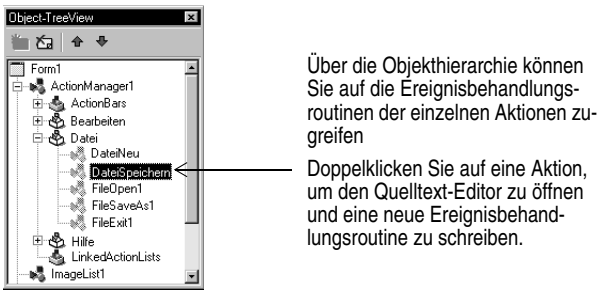
Um eine Ereignisbehandlungsroutine für den Befehl *Speichern* zu erstellen, führen Sie folgende Arbeitsschritte aus:

1 Drücken Sie *F12,* um das Formular zu aktivieren. Doppelklicken Sie auf die Komponente *ActionManager*.

2 Doppelklicken Sie auf die Aktion *Datei / Speichern*.

Der Quelltext-Editor wird geöffnet, und der Cursor befindet sich bereits innerhalb der Ereignisbehandlungsroutine.

Tip: Sie können auch in der Objekthierarchie auf die Aktion *Datei/DateiSpeichern* doppelklicken.

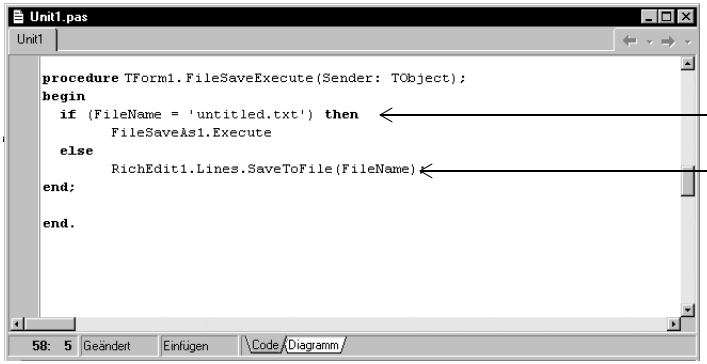


3 Geben Sie direkt an der Cursorposition (zwischen `begin` und `end`) die folgenden Zeilen ein:

```
if (FileName = 'unbenannt.txt') then
   FileSaveAs1.Execute
else
   RichEdit1.Lines.SaveToFile(FileName);
```

Dieser Code bewirkt, daß der Texteditor das Dialogfenster *Speichern unter* öffnet, wenn die Datei noch nicht benannt worden ist, so daß der Anwender einen Namen zuweisen kann. Andernfalls wird die Datei unter dem aktuellen Namen gespeichert. Das Dialogfenster *Speichern unter* wird im Abschnitt »Eine Ereignisbehandlungsroutine für den Befehl Speichern unter« auf Seite 4-22 definiert. *FileSaveAs1BeforeExecute* ist der automatisch generierte Name für den Befehl *Speichern unter*.

Wenn Sie fertig sind, sollte die Ereignisbehandlungsroutine folgenden Inhalt haben:



Das war's zum Befehl *Datei / Speichern.*

Eine Ereignisbehandlungsroutine für den Befehl Speichern unter

Bei Aufruf der *SaveDialog*-Methode *Execute* wird das Standarddialogfenster für das Speichern von Dateien unter neuem Namen aufgerufen. Um eine Ereignisbehandlungsroutine für den Befehl *Speichern unter* zu erstellen, führen Sie folgende Arbeitsschritte aus:

1. Drücken Sie *F12,* um das Formular zu aktivieren. Doppelklicken Sie auf die Komponente *ActionManager.*
2. Markieren Sie die Aktion *Datei / Speichern unter.*
3. Öffnen Sie die Registerkarte *Eigenschaften* des Objektinspektors. Standardmäßig vergibt Delphi den Namen *FileSaveAs1.Dialog.*
4. Klicken Sie im Objektinspektor neben der Eigenschaft *Dialog* auf das Pluszeichen, und legen Sie die folgenden Eigenschaften fest:
   - Setzen Sie *DefaultExt* auf `txt`.
   - Doppelklicken Sie auf den Textbereich neben *Filter,* um den Filter-Editor anzuzeigen. Legen Sie im Filter-Editor, wie beim Dialogfeld *Datei öffnen,* die Filter für die Dateiformate fest. Tragen Sie in die erste Zeile unter der Spalte *Filter Name* den Text `Textdateien (*.txt)` ein. Geben Sie in die Spalte *Filter* `*.txt` ein. Tragen Sie in die zweite Zeile unter der Spalte *Filter Name* den Text `Alle Dateien (*.*)` ein und in die Spalte *Filter* den Text *`*.*`*. Klicken Sie auf *OK.*
   - Tragen Sie neben *Title* den Titel `Speichern unter` ein.
5. Öffnen Sie im Objektinspektor die Registerkarte *Ereignisse.* Doppelklicken Sie in den Textbereich neben *BeforeExecute,* so daß `FileSaveAs1BeforeExecute` erscheint. Der Quelltext-Editor wird geöffnet, und der Cursor befindet sich im Quelltext-Editor.

6 Geben Sie direkt an der Cursorposition folgende Zeilen ein:

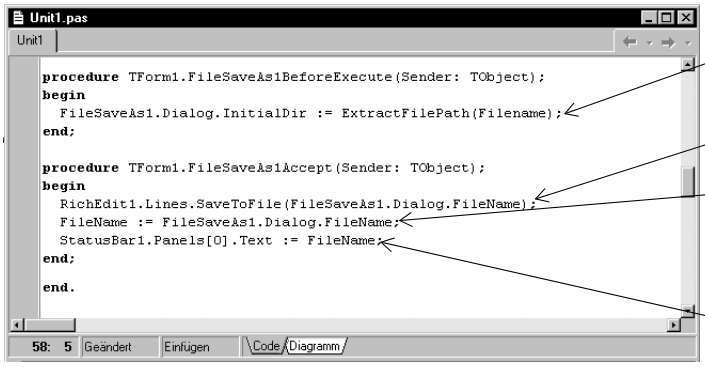
```
FileSaveAs1.Dialog.InitialDir := ExtractFilePath(Filename);
```

7 Im Objektinspektor sollte noch die Registerkarte *Ereignisse* geöffnet sein. Doppelklicken Sie in den Textbereich neben *OnAccept*, so daß `FileSaveAs1Accept` erscheint.

8 Der Quelltext-Editor wird geöffnet, und der Cursor befindet sich bereits innerhalb der Ereignisbehandlungsroutine. Geben Sie die folgenden Zeilen ein:

```
RichEdit1.Lines.SaveToFile(FileSaveAs1.Dialog.FileName);
FileName := FileSaveAs1.Dialog.FileName;
StatusBar1.Panels[0].Text := FileName;
```

Wenn Sie fertig sind, sollte die Ereignisbehandlungsroutine *FileSaveAs* folgenden Inhalt haben:



Diese Zeile richtet das auf das zuletzt aufgerufene Verzeichnis als das Standardverzeichnis ein

Diese Zeile speichert den Text in die angegebene Datei

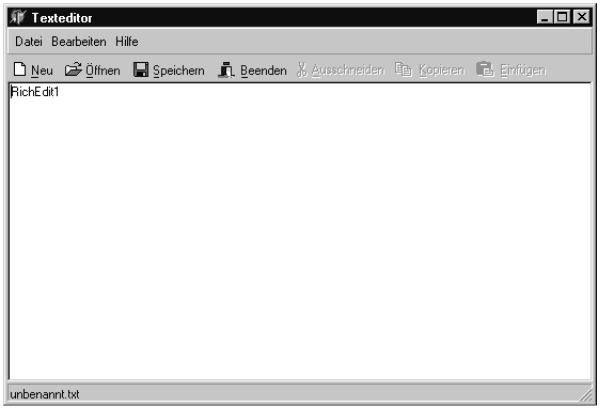
Hier wird der FileName des Hauptformulars auf den im Dialogfenster *SaveAs* angegebenen gesetzt.

Diese Zeile fügt den Dateinamen in die Statusleiste ein

Das war's zum Befehl *Datei / Speichern unter.*

9 Wählen Sie *Datei / Alles speichern,* damit das Projekt gespeichert wird.

10 Um zu sehen, was daraus geworden ist, führen Sie die Anwendung aus, indem Sie *F9* drücken.



Ihre Anwendung besitzt jetzt die volle Funktionalität Die Bilder erscheinen neben den Befehlen, mit denen Sie im Bildindex verknüpft wurden, und die Menüs, die Werkzeugleiste, der Textbereich und die Statusleiste werden auf dem Formular dargestellt..

Beachten Sie, das keine Namen für nichtvisuelle Komponenten angezeigt werden. Im Formular erscheinen die Menüs, die Symbolleiste, der Textbereich und die Statusleiste.

Die meisten der Schaltflächen und Befehlssymbole funktionieren, doch sind Sie noch nicht ganz fertig.

Wenn Sie im unteren Teil des Quelltext-Editors irgendwelche Fehlermeldungen erhalten, sollten Sie darauf klicken, damit die Stelle im Code angezeigt wird, an der der Fehler aufgetreten ist. Überprüfen Sie, ob Sie die Schritte in der Arbeitsanleitung genau befolgt haben.

11 Um den Entwurfsmodus wieder zu aktivieren, klicken Sie rechts oben im Laufzeit-Fenster auf das *X*.

Eine Hilfedatei erstellen

Es empfiehlt sich, zu jedem Programm eine Hilfedatei zu erstellen, die erläutert, wie die Anwendung funktioniert. Delphi stellt den Microsoft Help Workshop zur Verfügung, der sich im Verzeichnis C:\Programme\Borland\Delphi6\Help\Tools befindet. Dieser umfaßt auch Informationen zu Entwurf und Compilierung einer Windows-Hilfedatei. Um Beispieltexteditor kann der Anwender mit *Hilfe / Inhalt* oder *Hilfe / Index* eine Hilfedatei aufrufen, wobei entweder die Registerkarte *Inhalt* oder *Index* angezeigt wird.

Weiter vorne haben Sie im Aktions-Manager die Aktionen *HilfeInhalt* und *HilfeIndex* erstellt, um die Registerkarte *Inhalt* oder *Index* einer compilierten Hilfedatei anzuzeigen. Sie müssen den Hilfeparametern konstante Werte zuweisen und Ereignisbehandlungsroutinen erstellen, die das Gewünschte anzeigen.

Um die Hilfebefehle anwenden zu können, müssen Sie eine Windows-Hilfedatei erstellen und compilieren. Die Erstellung von Hilfedateien würde das Thema dieses Leitfadens sprengen. Dennoch können Sie als Beispiel eine RTF-Datei (TextEditor.rtf), eine Hilfedatei (TextEditor.hlp) und eine Inhaltsdatei (TextEditor.cnt) herunterladen:

1 Öffnen Sie die Datei D6X1.ZIP im Verzeichnis C:\Programme\Borland\Delphi6\Help.

2 Extrahieren und speichern Sie die HLP- und die CNT-Datei in Ihrem Texteditorverzeichnis, standardmäßig das Verzeichnis C:\Programme\Borland\Delphi6\Projects\Texteditor.

Hinweis: Sie können in Ihrem Projekt jede beliebige HLP- oder CNT-Datei verwenden (z. B. auch eine der Delphi-Hilfedateien zusammen mit der zugehörigen CNT-Datei). Damit sie von der Anwendung gefunden werden, müssen Sie sie in TextEditor.hlp und TextEditor.cnt umbenennen.

Eine Ereignisbehandlungsroutine für den Befehl Hilfe / Inhalt

Um eine Ereignisbehandlungsroutine für den Befehl *Hilfe / Inhalt* zu erstellen, führen Sie folgende Schritte aus:

1 Doppelklicken Sie auf die Komponente *ActionManager*.

2 Markieren Sie im Aktions-Manager-Editor die Kategorie *Hilfe*, und doppelklicken Sie dann auf die Aktion *HilfeInhalt*.

Der Quelltext-Editor wird geöffnet, und der Cursor befindet sich bereits innerhalb der Ereignisbehandlungsroutine.

3 Geben Sie direkt vor der Cursorposition (also vor `begin`) die folgenden Zeilen ein:

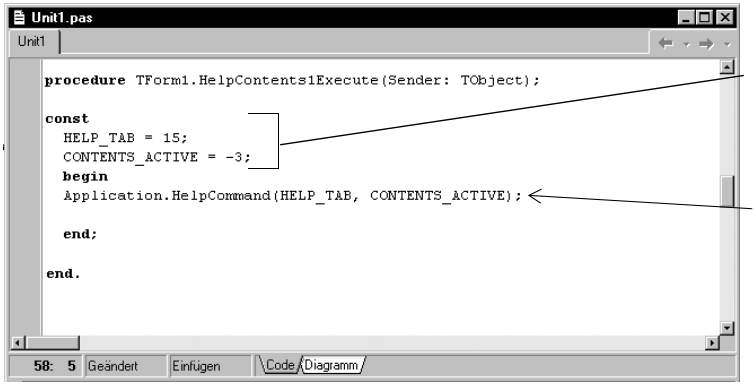
```
const
 HELP_TAB = 15;
 CONTENTS_ACTIVE = -3;
```

Direkt hinter `begin` geben Sie die folgende Zeile ein:

```
Application.HelpCommand(HELP_TAB, CONTENTS_ACTIVE);
```

Dieser Code weist den *HelpCommand*-Parametern konstante Werte zu. Indem Sie HELP_TAB auf 15 setzen, wird das Hilfedialogfenster angezeigt und indem Sie CONTENTS_ACTIVE auf -3 setzen, wird die Registerkarte *Inhalt* angezeigt.

Wenn Sie fertig sind, sollte die Ereignisbehandlungsroutine folgenden Inhalt haben:



Diese Zeilen definieren den Befehl und die Datenparameter der Methode *HelpCommand* der Anwendung *TApplication*.

Diese besagt, daß der Hilfedialog mit der Registerkarte *Inhalt* angezeigt werden soll.

Hinweis: Um Erläuterungen zum Ereignis *HelpCommand* aufzurufen, setzen Sie im Editor den Cursor neben `HelpCommand` und drücken *F1*.

Das war's zum Befehl *Hilfe / Inhalt*.

Eine Ereignisbehandlungsroutine für den Befehl Hilfe / Index

Um eine Ereignisbehandlungsroutine für den Befehl *Hilfe / Index* zu erstellen, führen Sie folgende Schritte aus:

1. Der Aktions-Manager-Editor sollte immer noch angezeigt werden. Falls nicht, doppelklicken Sie im Formular auf die Komponente *ActionManager*.
2. Markieren Sie im Aktions-Manager-Editor die Kategorie *Hilfe*, und doppelklicken Sie dann auf die Aktion *HilfeIndex*.

   Der Quelltext-Editor wird geöffnet, und der Cursor befindet sich bereits innerhalb der Ereignisbehandlungsroutine.
3. Geben Sie direkt vor der Cursorposition (also vor `begin`) die folgenden Zeilen ein:

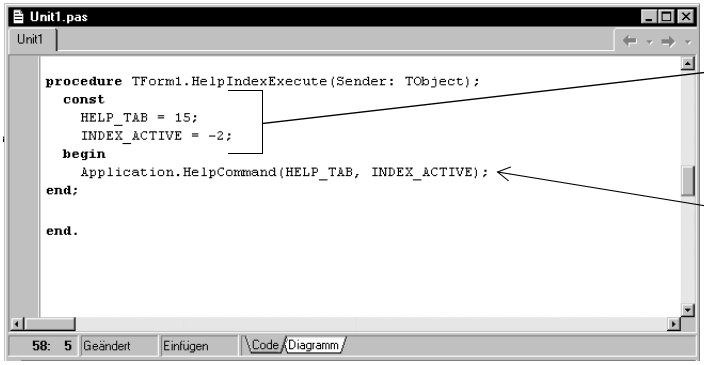
```
    const
  HELP_TAB = 15;
  INDEX_ACTIVE = -2;
```

   Direkt hinter `begin` geben Sie die folgende Zeile ein:

```
Application.HelpCommand(HELP_TAB, INDEX_ACTIVE);
```

   Dieser Code weist den *HelpCommand*-Parametern konstante Werte zu. Indem Sie HELP_TAB auf 15 setzen, wird das Hilfedialogfenster angezeigt und indem Sie CONTENTS_ACTIVE auf -2 setzen, wird die Registerkarte *Index* angezeigt.

   Wenn Sie fertig sind, sollte die Ereignisbehandlungsroutine folgenden Inhalt haben:



   Und das war's schon wieder zum Befehl *Hilfe / Index*.

Ein Info-Fenster erstellen

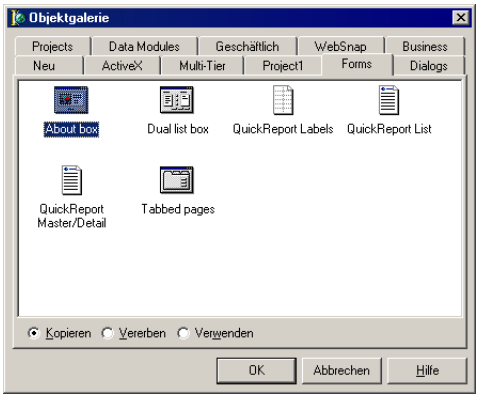
Viele Anwendungen besitzen ein Info-Fenster, das produktspezifische Informationen wie Name, Version, Logos anzeigt sowie eventuelle rechtliche Informationen wie das Copyright.

Im Aktions-Manager haben Sie bereits den Befehl *Hilfe / Info* eingerichtet.

Um ein Info-Fenster hinzuzufügen, führen Sie folgende Schritte aus:

1 Wählen Sie *Datei / Neu / Weitere*, um das Dialogfenster *Objektgalerie* einzublenden, und öffnen Sie die Registerkarte *Formulare.*

2 Doppelklicken Sie auf der Registerkarte *Formulare* auf *Aboutbox*.



Das Info-Fenster (About box) ist eines der in Delphi vordefinierten Formulare

Wenn *Kopieren* standardmäßig aktiviert ist, wird eine Kopie des Info-Fensters in Ihr Projekt eingefügt

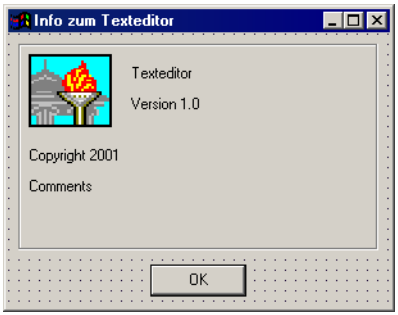
Es wird ein neues Formular angelegt, das die Erstellung eines Info-Fensters erleichtert.

3 Markieren Sie das Formular selbst (klicken Sie in den Gitterbereich) und tragen Sie im Objektinspektor für den Wert *Caption* den Text `Info zum Texteditor` ein.

4 Öffnen Sie die Registerkarte *Eigenschaften* des Objektinspektors und ändern Sie die *Caption*-Eigenschaften der folgenden *TLabel*-Einträge:

- Ändern Sie den `Product Name` auf `Texteditor`.
- Tragen Sie hinter *Version* den Wert `1.0` ein.

- Tragen Sie hinter `Copyright` das aktuelle Jahr ein.



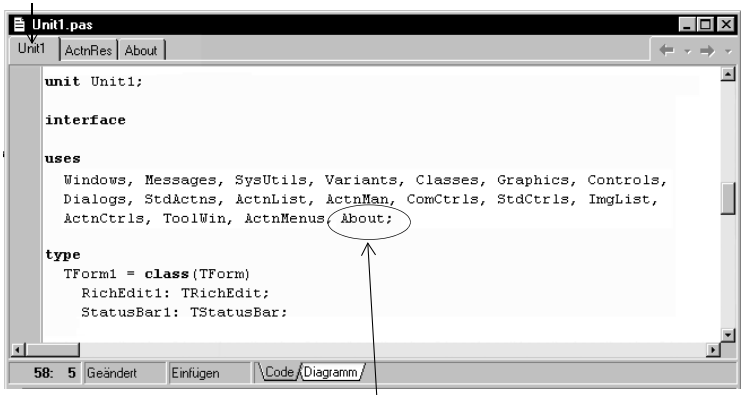
Die Objektablage enthält ein Standard-Info-Fenster, das Sie zur Beschreibung Ihrer Anwendung nach Bedarf abändern können.

5 Speichern Sie das Info-Fenster, indem Sie *Datei / Speichern unter* wählen und es unter dem Namen `Info.pas` speichern.

6 Im Delphi-Quelltext-Editor sollten nun drei Unit-Dateien angezeigt werden: Unit1, ActnRes und Info. Klicken Sie auf den Registerreiter *Unit1*, um Unit1.pas anzuzeigen. Die Unit *ActnRes* benötigen Sie nicht, doch können Sie sie einfach stehen lassen.

7 Öffnen Sie die Registerkarte *Unit1*, und fügen Sie die neue Unit *Info* hinzu, indem Sie in die Liste der aufgenommenen Units in der **uses**-Klausel `Info` eintragen.

Klicken Sie auf den Registerreiter, um die mit einer Unit verbundene Datei anzuzeigen. Wenn Sie während der Arbeit an einem Projekt weitere Dateien öffnen, erscheinen im Quelltext-Editor zusätzliche Registerreiter.



Wenn Sie ein neues Formular für Ihre Anwendung anlegen, so müssen Sie es in die **uses**-Klausel des Hauptformulars eintragen. Hier fügen Sie auch das Info-Fenster ein.

8 Drücken *F12* um den Entwurfsmodus wieder zu aktivieren. Doppelklicken Sie auf die Komponente *ActionManager*.

9 Doppelklicken Sie auf die Aktion *HilfeInfo,* um eine Ereignisbehandlungsroutine zu erstellen. Geben Sie direkt an der Cursorposition folgende Zeilen ein:

```
AboutBox.ShowModal;
```

Dieser Code öffnet das Hilfe-Fenster, wenn der Anwender *Hilfe / Info* wählt. *ShowModal* öffnet das Formular in einem modalen Zustand, d. h. der Anwender kann erst wieder im Texteditor arbeiten, wenn das Formular geschlossen ist.

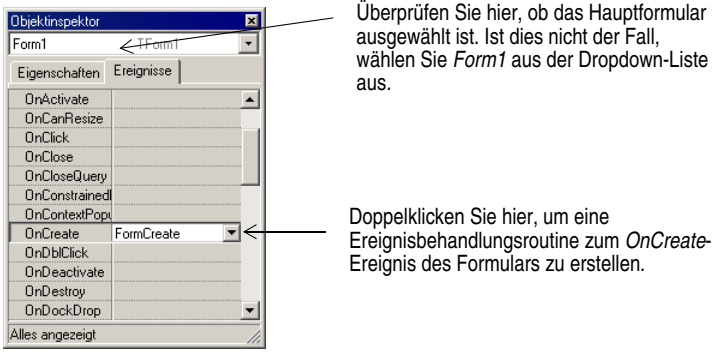
Die Anwendung fertigstellen

Die Anwendung ist nun fast fertig. Allerdings müssen Sie noch einige Objekte im Hauptformular festlegen. Um die Anwendung fertigzustellen, führen Sie folgende Schritte aus:

1 Drücken Sie *F12,* um das Hauptformular zu aktivieren.

2 Stellen Sie sicher, daß das Hauptformular selbst ausgewählt ist, und nicht eine seiner Komponenten. Dies ist der Fall, wenn in der Liste oben im Objektinspektor der Eintrag *Form1:* `TForm1` angezeigt wird. (Ist dies nicht der Fall, wählen Sie *Form1* aus der Dropdown-Liste aus.)

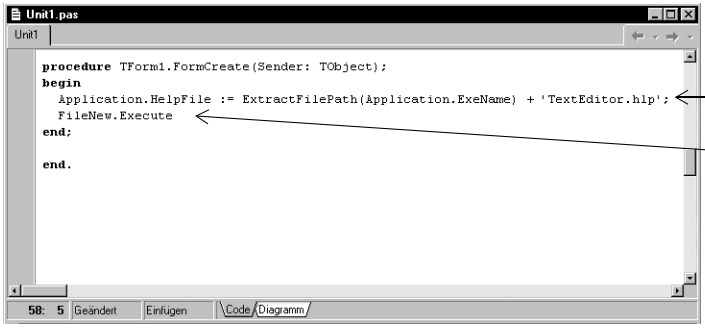
3 Öffnen Sie die Registerkarte *Ereignisse,* und wählen Sie neben *OnCreate* den Eintrag `FormCreate` aus der Dropdown-Liste aus, um eine Ereignisbehandlungsroutine zu erstellen, die festlegt, was beim Erstellen des Formulars (also beim Öffnen der Anwendung) passiert.



4 Geben Sie im Quelltext-Editor direkt an der Cursorposition folgende Zeilen ein:

```
Application.HelpFile := ExtractFilePath(Application.ExeName) + 'TextEditor.hlp';
FileNew.Execute
```

Dieser Code initialisiert die Anwendung, indem eine Hilfedatei zugewiesen, der Wert von *FileName* auf `unbenannt.txt` gesetzt, der Dateiname in die Statusleiste eingefügt und der Inhalt des Textbearbeitungsbereich gelöscht wird.



5 Wählen Sie *Datei / Alles speichern*, damit Ihre Änderungen gespeichert werden.

6 Drücken Sie *F9,* um die Anwendung auszuführen.

Herzlichen Glückwunsch! Sie haben es geschafft.

Kapitel

5

Die Umgebung anpassen

In diesem Kapitel erfahren Sie, wie Sie die Tools in der Entwicklungsumgebung von Delphi Ihren Bedürfnissen anpassen können.

Den Arbeitsbereich organisieren

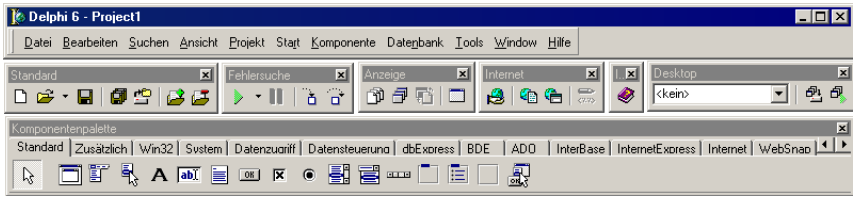
Die IDE stellt zahlreiche Tools zur Verfügung, die die Anwendungsentwicklung vereinfachen. Bei einer solchen Fülle an Tools ist es ratsam, den Arbeitsbereich möglichst gut zu organisieren. Dazu gehört das Anordnen von Menüs und Symbolleisten, das Kombinieren von Tool-Fenstern und das Speichern neuer Desktop-Layouts.

Menüs und Symbolleisten anordnen

Im Hauptfenster können Sie die Menüs, die Symbolleisten und die Komponentenpalette beliebig anordnen, indem Sie die Griffleiste auf der linken Seite anklicken und das jeweilige Element auf eine neue Position ziehen.

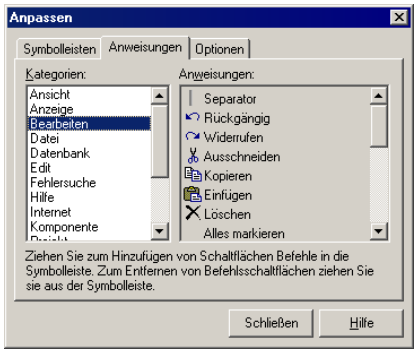
Menüs und Symbolleisten lassen sich innerhalb des Hauptfensters verlagern. Sie verlagern eine bestimmte Symbolleiste, indem Sie an der Griffleiste (dem Doppelbalken am linken Ende) ziehen

Sie können diese Elemente vom Hauptfenster trennen, beliebig anordnen oder ganz vom Desktop entfernen. Bei einer Konfiguration mit zwei Monitoren ist dies besonders praktisch.



Ein umgestaltetes Hauptfenster.

Sie können Tools zu Symbolleisten hinzufügen oder davon entfernen, indem Sie *Ansicht* / *Symbolleisten* / *Anpassen* wählen. Anschließend öffnen Sie die Registerkarte *Anweisungen,* wählen eine Kategorie und dann eine Anweisung aus und ziehen diese auf die Stelle der Symbolleiste, an der die zugehörige Schaltfläche eingesetzt werden soll.



Wählen Sie auf der Registerkarte *Anweisungen* eine Anweisung aus und ziehen Sie sie auf eine der Symbolleisten.

Aktivieren Sie auf der Registerkarte *Optionen* das Kontrollfeld *Kurzhinweise anzeigen,* damit zu den Komponenten und Befehlssymbolen die Kurzhinweise eingeblendet werden

Weitere Informationen

Suchen Sie im Hilfeindex nach dem Eintrag »Symbolleisten, anpassen«.

Tool-Fenster andocken

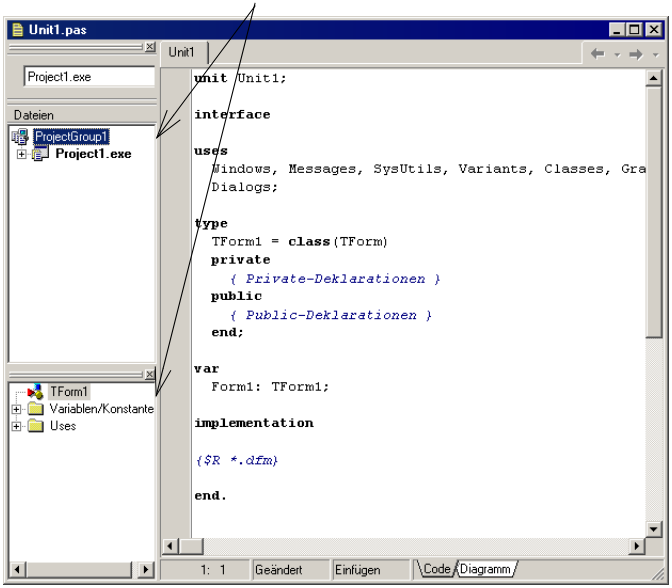
In Delphi können die verschiedenen Tool-Fenster beliebig auf dem Desktop angeordnet und unabhängig voneinander geöffnet und geschlossen werden. Viele dieser Fenster können auch zur besseren Übersicht an andere Fenster *angedockt* werden. Darunter ist zu verstehen, daß mehrere Fenster fest miteinander verbunden werden können, so daß sie eine Einheit bilden und zusammen verlagert werden können. Diese Möglichkeiten verhelfen Ihnen zu einer effizienten Nutzung der verfügbaren Bildschirmfläche und zu einem raschen Zugriff auf die betreffenden Tools.

Über das Menü *Ansicht* können Sie jedes Tool-Fenster öffnen und direkt aneinander andocken. Dazu ein Beispiel: Wenn Sie Delphi zum ersten Mal in der Standardkonfiguration aufrufen, ist auf der linken Seite des Quelltext-Editors der Code-Explorer

angedockt. Bei Bedarf können Sie an diese beiden Fenster auch die Projektverwaltung andocken.

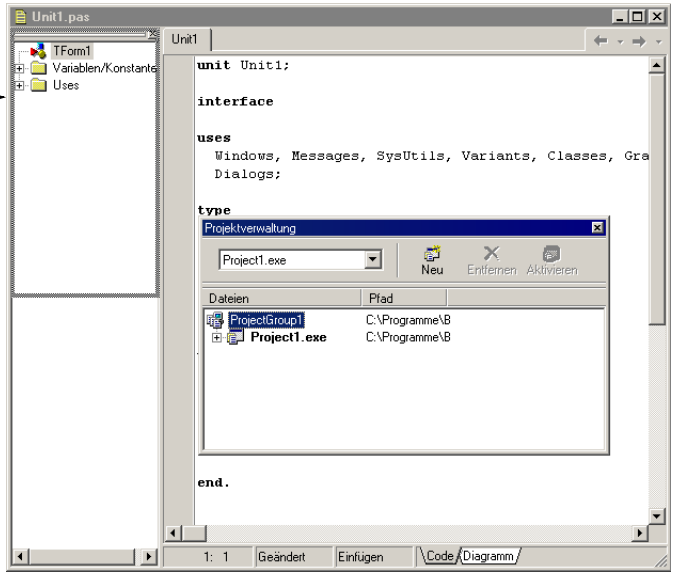
Hier sind die Projektverwaltung und der Code-Explorer an den Quelltext-Editor angedockt

Beim Andocken erhalten Sie entweder Fenster mit Griffleisten (wie hier) oder mit mehreren Registerkarten (wie auf Seite 5–4)

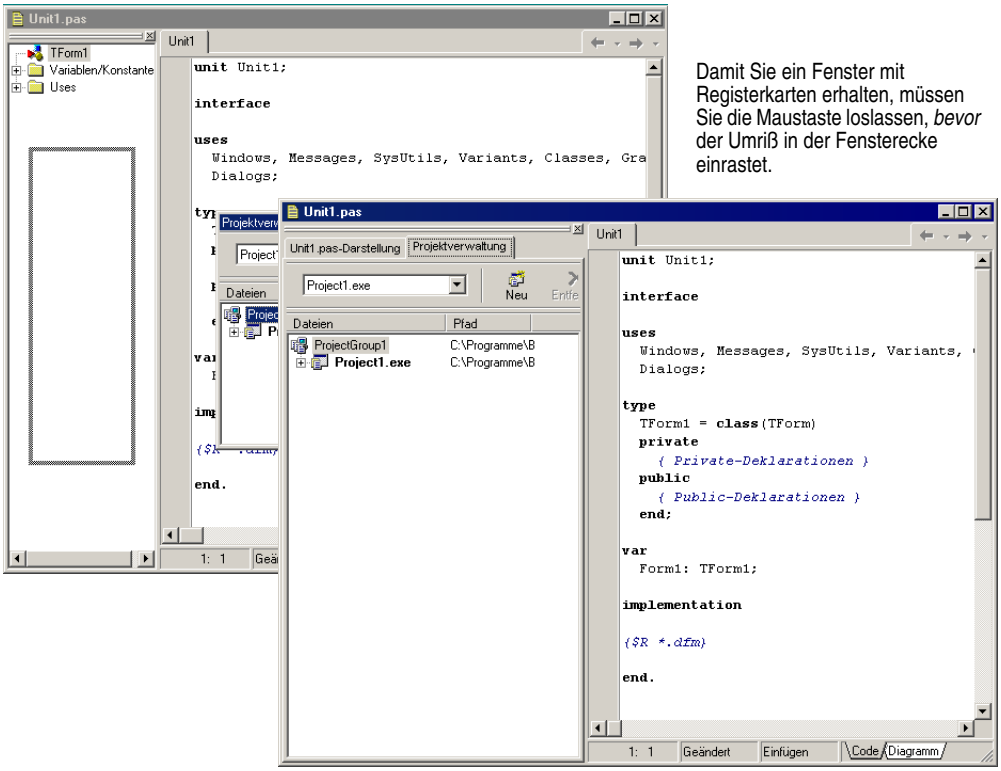


Um ein Fenster anzudocken, klicken Sie auf seine Titelleiste, und ziehen es über das andere Fenster. Wenn sich sein Umriß in ein kleines Rechteck verwandelt und in einer Ecke einrastet, lassen Sie die Maustaste los. Die beiden Fenster sind aneinander angedockt.

Damit Sie Fenster mit Griffleisten erhalten, lassen Sie die Maustaste los, sobald der Umriß in der Fensterecke einrastet.



Sie können durch das Andocken von Tools auch ein Fenster mit Registerkarten bilden.



Damit Sie ein Fenster mit Registerkarten erhalten, müssen Sie die Maustaste loslassen, *bevor* der Umriß in der Fensterecke einrastet.

Um eine Andockung eines Fensters aufzuheben, doppelklicken Sie auf seinen Griff oder auf sein Register.

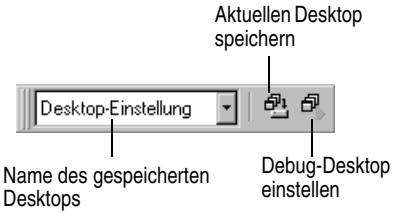
Um das automatische Andocken zu verhindern, drücken entweder beim Ziehen eines Fensters die Taste *Strg* oder wählen *Tools / Umgebungsoptionen*, öffnen die Registerkarte *Präferenzen*, und deaktivieren die Option *Automatisch Andocken beim Ziehen*.

Weitere Informationen

Suchen Sie im Hilfeindex nach dem Begriff »Andocken«.

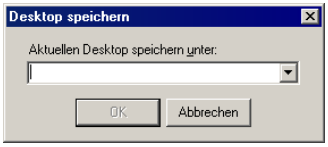
Desktop-Layouts speichern

Delphi gibt Ihnen die Möglichkeit, das Desktop-Layout an Ihre individuellen Anforderungen anzupassen und zu speichern. Dazu steht in der IDE eine Desktop-Symbolleiste mit einer Liste der vorhandenen Desktop-Einstellungen und zwei Befehlssymbolen zur Verfügung.



Richten Sie den Desktop in der gewünschten Form ein. Dazu gehört sein Aussehen, die Größe der einzelnen Elemente und die Andockpositionen einzelner Fenster.

Zuletzt klicken Sie in der Symbolleiste *Desktop* auf das Symbol *Aktuellen Desktop speichern* oder wählen *Ansicht* / *Desktops* / *Desktop speichern*, und geben einen Namen für das neue Layout ein.



Geben Sie einen Namen für das zu speichernde Desktop-Layout ein, und klicken Sie auf *OK*

Weitere Informationen

Suchen Sie im Hilfeindex nach dem Begriff »Desktop-Layout«.

Die Komponentenpalette anpassen

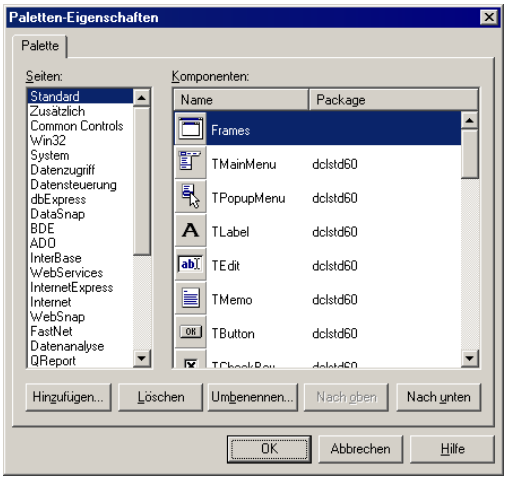
In der Standardkonfiguration sind die in der Komponentenpalette enthaltenen VCL- oder CLX- Objekte, geordnet nach ihren jeweiligen Funktionen, auf verschiedene Registerkarten verteilt. Sie haben folgende Möglichkeiten zur Anpassung der Komponentenpalette:

- Sie können Komponenten ausblenden und anders anordnen.
- Sie können Registerkarten hinzufügen, entfernen, neu anordnen oder auch umbenennen.
- Sie können Komponentenvorlagen erstellen und zur Palette hinzufügen.
- Sie können zusätzliche Komponenten installieren.

Die Komponentenpalette umgestalten

Um Registerkarten hinzuzufügen, zu entfernen und neu anzuordnen oder um Komponenten auszublenden oder ihre Reihenfolge in den einzelnen Registerkarten zu ändern, benutzen Sie das Dialogfenster *Paletteneigenschaften*. Um dieses Dialogfenster zu öffnen, gibt es verschiedene Möglichkeiten:

- Wählen Sie *Komponente* / *Palette konfigurieren*.
- Wählen Sie *Tools* / *Umgebungsoptionen*, und öffnen Sie die Registerkarte *Palette*.
- Klicken Sie die Komponentenpalette mit der rechten Maustaste an, und wählen Sie *Eigenschaften*.



Sie können die Palette umgestalten und neue Registerkarten hinzufügen.

Weitere Informationen

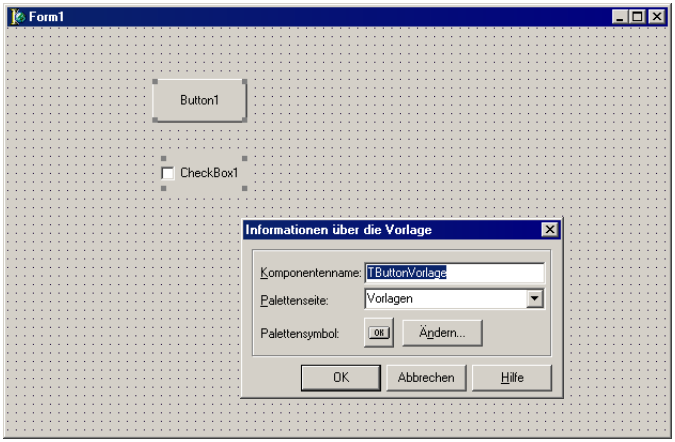
Klicken Sie im Dialogfenster *Paletteneigenschaften* auf die Schaltfläche *Hilfe*.

Komponentenvorlagen erstellen

Komponentenvorlagen sind Gruppen von Komponenten, die Sie in nur einem Arbeitsschritt in ein Formular einfügen. Sie geben Ihnen die Möglichkeit, mehrere Komponenten in einem einzelnen Formular einzurichten und dann ihre Anordnung, ihre Standardeigenschaften und Ereignisbehandlungsroutinen zur Wiederverwendung in anderen Formularen als Verbund zu speichern.

Um eine Komponentenvorlage zu erstellen, plazieren Sie einfach die gewünschten Komponenten in einem Formular, legen im Objektinspektor ihre Eigenschaften fest und markieren anschließend die gesamte Komponentengruppe, indem Sie mit der Maus ein Markierungsfeld darüber ziehen. Wählen Sie dann *Komponente* / *Komponentenvorlage erzeugen*. Wenn sich das Dialogfenster *Information über die Vorlage* öffnet, können Sie dann für die Vorlage einen Namen auswählen und angeben, auf welcher Registerkarte der Komponentenpalette sie erscheinen soll. Ferner können Sie der Vorlage ein Symbol zuordnen, das später in der Komponentenpalette angezeigt wird.

Nachdem eine solche Vorlage in ein Formular eingefügt wurde, können Sie die einzelnen Komponenten unabhängig voneinander neu positionieren, ihre Eigenschaften ändern und Ereignisbehandlungsroutinen erstellen oder modifizieren – gerade so, als hätten Sie die Komponenten einzeln in das Formular aufgenommen.



Weitere Informationen

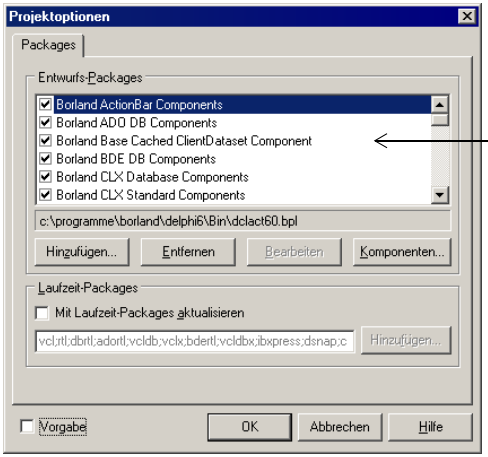
Suchen Sie im Hilfeindex nach dem Begriff »Komponentenvorlagen«.

Komponenten-Packages installieren

Unabhängig davon, ob Sie eigene Komponenten erstellen oder diese von einem Hersteller erwerben, müssen diese erst in ein *Package* kompiliert werden, bevor Sie diese in der Komponentenpalette installieren können.

Bei einem Package handelt es sich um eine spezielle DLL mit Quelltext, der von mehreren Delphi-Anwendungen, der IDE oder beiden gemeinsam genutzt werden kann. *Laufzeit-Package*s liefern Funktionalität, wenn Benutzer eine Anwendung ausführen. *Entwurfszeit-Packages* dienen zur Installation von Komponenten in der IDE. Delphi-Packages haben die Erweiterung .BPL.

Wenn die Komponenten eines Fremdanbieters bereits in ein Package compiliert sind, folgen Sie entweder den Anweisungen des Anbieters, oder wählen den Befehl *Komponente / Packages installieren*.



Diese Komponenten sind in Delphi vorinstalliert. Wenn Sie neue Komponenten eines Fremdanbieters installieren, erscheinen sie ebenfalls in dieser Liste

Klicken Sie auf *Komponenten*, um zu sehen, welche Komponenten ein Package enthält

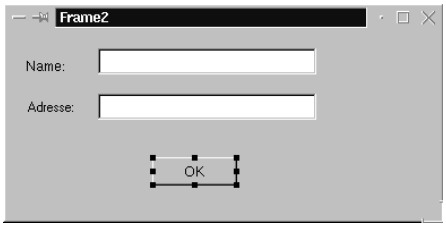
Weitere Informationen

Suchen Sie im Hilfeindex nach den Begriffen »Komponenten installieren« und »Packages«.

Frames

Ein Frame (*TFrame*) ist, wie auch ein Formular, ein Container für wiederverwendbare Komponenten. Allerdings ähnelt ein Frame eher einer angepaßten Komponente als einem Formular. Um den mehrfachen Einsatz zu erleichtern, können Frames in der Komponentenpalette gespeichert und in Formularen, anderen Frames oder sonstigen Container-Objekten verschachtelt werden. Nachdem ein Frame erstellt und gespeichert worden ist, funktioniert er weiterhin als Unit und vererbt Änderungen weiterhin an die Komponenten, die er beinhaltet (einschließlich anderer Frames). Und ist ein Frame in einen anderen Frame oder in ein Formular eingebettet, so erbt er nach wie vor Änderungen an dem Frame, von dem er abgeleitet ist.

Um einen neuen Frame zu öffnen, wählen Sie *Datei / Neu / Frame*.



In einen Frame können Sie alle visuellen und nicht visuellen Komponenten einfügen, die Sie benötigen. In den Quelltext-Editor wird automatisch eine neue Unit eingefügt

Weitere Informationen

Suchen Sie im Hilfeindex nach den Begriffen »Frames« und »Tframe«.

ActiveX-Steuerelemente hinzufügen

Sie können auch ActiveX-Steuerelement in die Komponentenpalette einbinden und sie in Ihren Delphi-Projekten verwenden. Dazu wählen Sie *Komponente / ActiveX importieren*. Im Dialogfenster *ActiveX importieren* können Sie neue ActiveX-Steuerelemente registrieren oder ein bereits registriertes Steuerelement für die Installation in der IDE auswählen. Wenn Sie ein ActiveX-Steuerelement in Delphi installieren, wird dafür eine »Wrapper«-Unit-Datei angelegt.

Weitere Informationen

Wählen Sie *Komponente / ActiveX importieren*, und klicken Sie auf die Schaltfläche *Hilfe*.

Projektoptionen festlegen

Wenn Sie Projektverzeichnisse verwalten und Projektoptionen für Formulare, Anwendungen, Compiler und Linker angeben müssen, wählen Sie *Projekt* / *Optionen*. Nehmen Sie im Dialogfenster *Projektoptionen* Änderungen vor, gelten diese zunächst nur für das aktuelle Projekt. Sie haben jedoch die Möglichkeit, Ihre Einstellungen als Vorgabe für neue Projekte zu speichern.

Standard-Projektoptionen festlegen

Um Ihre Auswahl als Standardeinstellungen für alle neuen Projekte zu speichern, aktivieren Sie links unten im Dialogfenster *Projektoptionen* das Kontrollfeld *Vorgabe*. Die aktuellen Einstellungen des Dialogfensters werden dann in die Optionendatei DEFPROJ.KOF im Verzeichnis Delphi6\Bin geschrieben. Um die Standardeinstellungen von Delphi wiederherzustellen, müssen Sie die Datei DEFPROJ.KOF löschen oder umbenennen.

Weitere Informationen

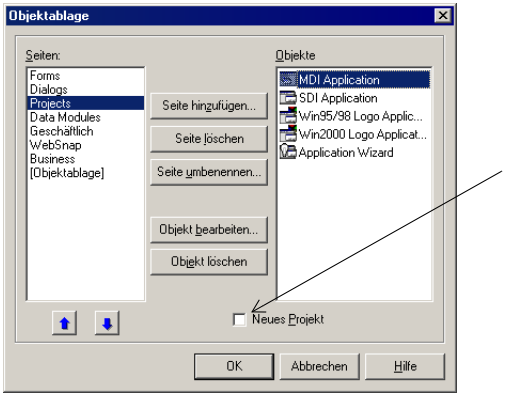
Suchen Sie im Hilfeindex nach dem Begriff »Projektoptionen (Dialogfenster)«.

Projekt- und Formularvorlagen als Standard festlegen

Wenn Sie *Datei / Neu / Anwendung* wählen, erstellt Delphi normalerweise eine neue Standardanwendung mit leerem Formular. Sie haben jedoch auch die Möglichkeit, eine Projekt*vorlage* als *Standardprojekt* zu definieren. Auf der Registerkarte *Projekte* speichern Sie ein eigenes Projekt als Vorlage in der Objektablage, indem Sie den Befehl *Projekte* / *Der Objektablage hinzufügen* wählen (siehe »Vorlagen in die Objektablage einfügen« auf Seite 5-10). Oder Sie verwenden eine der vordefinierten Delphi-Projektvorlagen aus der Objektablage aus (siehe »Die Objektablage« auf Seite 2-6).

Um ein Projekt als Projektvorlage festzulegen, wählen Sie *Tools* / *Objektablage*. Im Dialogfenster *Objektablage* klicken Sie in der Liste *Seiten* auf den Eintrag *Projekte*.

Wenn Sie ein Projekt auf der Registerkarte *Projekte* als Vorlage gespeichert haben, wird dieses in der Liste *Objekte* angezeigt. Wählen Sie den Namen der Vorlage aus, aktivieren Sie das Kontrollfeld *Neues Projekt* und klicken Sie auf *OK*.



Die Seiten der Objektablage enthalten entweder nur Projektvorlagen oder nur Formularvorlagen, oder eine Kombination aus beidem

Um eine Projektvorlage als Standard festzulegen, markieren Sie einen Eintrag in der Liste *Objekte* und aktivieren die Option *Neues Projekt*

Um eine Projektvorlage als Standard festzulegen, markieren Sie einen Eintrag in der Liste *Objekte* und aktivieren die Option *Neues Projekt*

Sobald Sie eine Projektvorlage als Vorgabe festgelegt haben, wird diese bei Auswahl des Befehls *Datei / Neu / Anwendung* automatisch geöffnet.

Auf dieselbe Weise können Sie auch aus der Liste der vorhandenen Formularvorlagen in der Objektablage ein *Standardformular für neue Formulare* und ein *Standard-Hauptformular* auswählen. Das Standardformular für neue Formulare wird verwendet, wenn Sie mit *Datei / Neu / Formular* ein neues Formular ins Projekt aufnehmen. Das Standard-Hauptformular ist das Formular, das zu Beginn einer neuen Anwendung erstellt wird. Wenn Sie kein Standardformular festgelegt haben, wird ein leeres Formular verwendet.

Natürlich haben Sie immer die Möglichkeit, Standardprojekte oder Standardformulare zu überschreiben. Dazu wählen Sie *Datei / Neu / Weitere* und treffen im Dialogfenster *Objektgalerie* die gewünschte Auswahl.

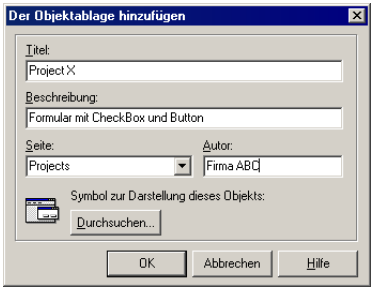
Weitere Informationen

Suchen Sie im Hilfeindex nach den Begriffen »Vorlagen, zur Objektablage hinzufügen«, »Projekte, Standardobjekte festlegen« und »Formulare, Standardformulare festlegen«.

Vorlagen in die Objektablage einfügen

Sie können eigene Objekte als *Vorlagen* in die Objektablage einfügen und anderen Entwicklern über ein Netzwerk zur Verfügung stellen. Die mehrfache Verwendung von Objekten hat den Vorteil, daß Sie Programmfamilien mit gemeinsamer Bedieneroberfläche und Funktionalität erstellen können. Dadurch sparen Sie nicht nur Entwicklungszeit, sondern verbessern auch die Qualität Ihrer Produkte.

Um etwa ein Projekt als Vorlage in die Objektablage einzufügen, speichern Sie zunächst das Projekt, wählen dann *Projekt / Der Objektablage hinzufügen* und füllen das gleichnamige Dialogfenster aus.



Geben Sie einen Titel, eine Beschreibung und den Namen des Autors ein. Wählen Sie in der Dropdown-Liste *Seite* den Eintrag *Projekte* aus, so daß Ihr Projekt in den Registerkarte *Projekte* der Objektablage erscheint.

Wenn Sie das Dialogfenster *Objektgalerie* das nächste Mal öffnen, erscheint Ihre Projektvorlage auf der Registerkarte *Projekte* (bzw. auf der Registerkarte, auf der Sie sie gespeichert haben). Wie Sie eine eigene Vorlage als Standard beim Öffnen von Delphi festlegen, erfahren Sie im Abschnitt »Projekt- und Formularvorlagen als Standard festlegen« auf Seite 5-9.

Weitere Informationen

Suchen Sie im Hilfeindex nach dem Begriff »Vorlagen, zur Objektablage hinzufügen«.

Einstellungen für Tools festlegen

In Delphi können Sie zahlreiche Aspekte steuern, die das Aussehen und die Funktionsweise der IDE betreffen, z. B. den Formular-Designer, Objektinspektor oder den Code-Explorer. Diese Einstellungen betreffen nicht nur das aktuelle Projekt, sondern auch solche, die zu einem späteren Zeitpunkt geöffnet und compiliert werden. Um die globalen IDE-Einstellungen für alle Projekte zu ändern, wählen Sie *Tools* / *Umgebungsoptionen*.

Weitere Informationen

Klicken Sie auf den verschiedenen Registerkarten des Dialogfensters *Umgebungsoptionen* die Schaltfläche *Hilfe* an, oder suchen Sie im Hilfeindex nach dem Begriff »Umgebungsoptionen (Dialogfenster)«.

Den Formular-Designer anpassen

Auf der Registerkarte *Designer* des Dialogfensters *Umgebungsoptionen* werden Einstellungen vorgenommen, die den Formular-Designer betreffen. Dort können Sie beispielsweise die Ausrichtfunktion des Rasters aktivieren oder deaktivieren, wodurch Komponenten an der nächsten Gitterlinie ausgerichtet werden, oder Sie können Namen oder *Titel* nicht visueller Komponenten im Formular anzeigen lassen oder verbergen.

Weitere Informationen

Öffnen Sie im Dialogfenster *Umgebungsoptionen* die Registerkarte *Designer*, und klikken Sie auf die Schaltfläche *Hilfe*.

Den Quelltext-Editor anpassen

Ein Tool, das in der Regel sofort angepaßt wird, ist der Quelltext-Editor. Die entsprechenden Bearbeitungsoptionen für Quelltext sind auf verschiedenen Registerkarten des Dialogfensters *Editor-Optionen* zu finden. So können Sie beispielsweise bestimmte Tastenbelegungen auswählen und Schriftarten, Randeinstellungen, Farben, Syntaxhervorhebungen, Tabulatoren sowie Einzugsarten festlegen.

Auch die Programmierhilfe-Tools, auf die Sie im Editor mittels der Registerkarte *Programmierhilfe* des Dialogfensters *Editor-Optionen* zugreifen können, lassen sich individuell konfigurieren. Diese Tools sind im Abschnitt »Die Programmierhilfen« auf Seite 2-8 beschrieben.

Weitere Informationen

Klicken Sie in den Registerkarten *Allgemein*, *Anzeige*, *Tastaturbelegung*, *Farben* und *Programmierhilfe* des Dialogfensters *Editor-Optionen* auf die Schaltfläche *Hilfe*.

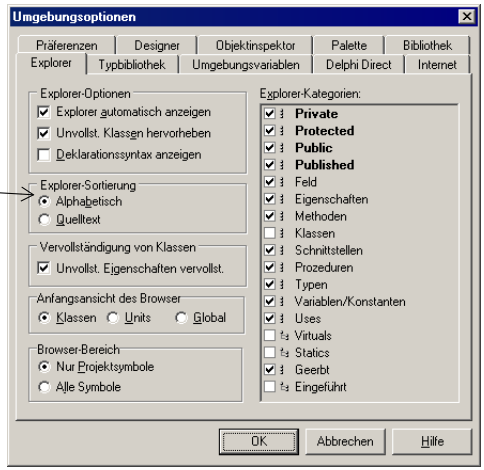
Den Code-Explorer anpassen

Beim Start von Delphi wird der Code-Explorer (siehe »Der Code-Explorer« auf Seite 2-13) automatisch geöffnet. Sie können dies verhindern, indem Sie *Tools* / *Umgebungsoptionen* wählen, und auf der Registerkarte *Explorer* die Option *Explorer automatisch anzeigen* deaktivieren.

Wie der Inhalt des Code-Explorers gruppiert wird, bestimmen Sie, indem Sie im lokalen Menü des Code-Explorers *Eigenschaften* wählen und unter *Explorer-Kategorien* die entsprechenden Kontrollfelder aktivieren bzw. deaktivieren. Ist eine Kategorie aktiviert, erscheinen die Elemente in dieser Kategorie unter demselben Knoten. Ist eine Kategorie nicht aktiviert, so ist jedes ihrer Elemente vom Diagrammstrang abgesetzt.

Angenommen, Sie haben die Kategorie *Publiziert* deaktiviert, so verschwindet zwar der Ordner *Publiziert*, nicht aber die darin enthaltenen Elemente.

Im Code-Explorer können Sie alle Quellelemente entweder alphabetisch sortieren lassen oder in der Reihenfolge, in der sie in der Quelldatei deklariert wurden



Damit für die verschiedenen Arten von Quellelementen im Code-Explorer der Ordner angezeigt wird, aktivieren Sie die entsprechende Explorer-Kategorie

Weitere Informationen

Suchen Sie im Hilfeindex nach dem Begriff »Code-Explorer, Umgebungsoptionen«.

Index

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z