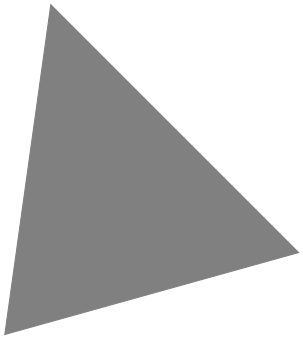
Einführung



Borland®
Delphi™ 5
für Windows 95, Windows 98 & Windows NT

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

Deutsche Ausgabe © 1999 Inprise 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

HDE1350GE21000

Inhalt

Kapitel 1
Einführung 1-1
Was ist Delphi? 1-1
Wo Sie Informationen finden 1-1
Online-Hilfe 1-2
Gedruckte Dokumentation 1-4
Inprise Developer Support Services 1-4
Typographische Konventionen 1-4

Kapitel 2
Die Umgebung kennenlernen 2-1
Delphi starten 2-1
Mit Symbolleisten, Menüs und Tastenkürzeln arbeiten 2-3
Komponenten in einem Formular plazieren . . 2-4
Aussehen und Verhalten von Komponenten ändern 2-5
Ereignisse auswählen 2-6
Quelltext anzeigen und bearbeiten 2-7
Formulardateien anzeigen. 2-7
Der Editor als Browser. 2-8
Der Code-Explorer 2-8
Projektverwaltung 2-9
Elemente und Struktur eines Projekts anzeigen 2-10
To-DoListen erstellen 2-11
Datenmodule entwerfen 2-11
Projekt- und Umgebungsoptionen einstellen . .2-12
Hilfe 2-14
Hilfe beim Programmieren 2-15
Vervollständigung von Klassen 2-16
Fehlersuche in Anwendungen 2-17
Datenbanken abfragen 2-18
Objekte als Vorlagen speichern 2-19

Kapitel 3
Die erste Anwendung – eine kurze Einführung 3-1
Eine neue Anwendung beginnen 3-1
Eigenschaftswerte festlegen 3-3
Objekte zum Formular hinzufügen 3-3
Auf eine Datenbank zugreifen 3-5
Symbolleisten- und Menüunterstützung hinzufügen 3-8
Ein Menü hinzufügen 3-9
Eine Symbolleiste hinzufügen 3-11
Eine Grafik anzeigen 3-12
Text- und Memoobjekte hinzufügen 3-14
Eine Ereignisbehandlungsroutine erstellen . . 3-16

Kapitel 4
Die Umgebung anpassen 4-1
Den Arbeitsbereich organisieren 4-1
Tool-Fenster andocken 4-1
Menüs und Symbolleisten organisieren. 4-4
Desktop-Einstellungen anpassen 4-5
Standard-Projektoptionen festlegen 4-5
Standardprojekte und -formulare festlegen . . .4-5
Einstellungen für Tools festlegen 4-7
Den Quelltexteditor anpassen 4-7
Den Formular-Designer anpassen 4-7
Optionen für den Code-Explorer festlegen . . 4-7
Die Komponentenpalette anpassen 4-8
Die Komponentenpalette anders konfigurieren. 4-9
Komponenten hinzufügen 4-10
ActiveX-Steuerelemente hinzufügen . . 4-10
Komponentenschablonen erstellen 4-10
Das Hilfesystem anpassen 4-11

Kapitel 5
Mit Delphi programmieren 5-1
Die Tools der Entwicklungsumgebung. 5-1
Die VCL verwenden 5-1
Exceptions behandeln 5-2
Datenbank-Connectivity und Datenbank-Utilities 5-3
BDE-Administrator 5-3
SQL-Explorer (Datenbank-Explorer) . . . 5-4
Datenbank-Desktop 5-4
Daten-Dictionary. 5-4
Projektarten 5-4
Anwendungen und Server 5-4
DLLs 5-5
Benutzerdefinierte Komponenten und Packages 5-5
Frames 5-5
COM und ActiveX. 5-6
Typbibliotheken 5-6
Anwendungen vertreiben 5-6
Anwendungen internationalisieren. 5-7

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.

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 95, Windows 98 und Windows NT erstellen können. Delphi stellt alle Tools zur Verfügung, die Sie benötigen, um Anwendungen zu entwickeln, zu testen, zu debuggen und zu vertreiben. Dazu gehören eine umfangreiche Bibliothek wiederverwendbarer Komponenten, eine Reihe von Entwurfs-Tools, Anwendungs- und Formularvorlagen sowie Programmierexperten. All diese Werkzeuge vereinfachen die Erstellung von Anwendungs-Prototypen und verkürzen die Entwicklungszeit.

Wo Sie Informationen finden

Für Delphi stehen vielerlei Informationsquellen zur Verfügung:

- Online-Hilfe
- Gedruckte Dokumentation
- Inprise Developer Support Services
- Die Web Sites Inprise.com und Borland.com

Informationen über die neuen Features dieser Programmversion finden Sie in der Online-Hilfe unter »Neuerungen« und auf der Web Site Borland.com.

Online-Hilfe

In der Online-Hilfe lassen sich detaillierte Informationen über die Features der Benutzeroberfläche, über die Sprachimplementierung, über Programmiertechniken sowie über die in der Visual Components Library (VCL) enthaltenen Komponenten abrufen. Zu Delphi gehören die in der folgenden Tabelle aufgeführten zentralen Hilfedateien:

Tabelle 1.1 Dateien der Online-Hilfe

Hilfedatei	Inhalt	Zielgruppe
Neuerungen in Delphi (Del5new.hlp)	Führt in die neuen Features der aktuellen Delphi-Version ein Enthält Links zu Detailinformationen.	Entwickler, die schon mit einer älteren Delphi-Version gearbeitet haben.
Delphi verwenden (Delphi5.hlp)	Führt in die Entwicklungsumgebung von Delphi ein und erläutert die Arbeit mit Formularen, Projekten und Packages. Außerdem werden die Grundlagen der komponentengestützten objektorientierten Programmierung behandelt.	Delphi-Einsteiger und Entwickler, die Fragen zur IDE haben
VCL-Referenz (Del5vcl.hlp)	Enthält eine detaillierte Referenz der VCL-Klassen, globalen Routinen, Typen und Variablen. Zu den einzelnen Objekte ist jeweils die Unit angegeben, in der sie deklariert sind, sowie ihre Position in der Objekthierarchie. Ferner werden alle verfügbaren Eigenschaften und Methoden beschrieben und praktische Anwendungsbeispiele gegeben.	Alle Delphi-Entwickler
Programmieren mit Delphi (Del5prog.hlp)	Enthält Details über die Verwendung der VCL-Komponenten und erläutert alltägliche Programmieraufgaben, wie z.B. die Behandlung von Exceptions, das Erstellen von Symbolleisten und Drag&Drop-Steuerelementen sowie die Verwendung von Grafiken.	Alle Delphi-Entwickler
Datenbankanwendungen entwickeln (Del5dbd.hlp)	Erläutert den Aufbau von ein- und mehrschichtigen Datenbankanwendungen und bietet Hintergrundinformationen über Datenbankarchitekturen, Datenmengen, Felder, Tabellen, Abfragen und Entscheidungskomponenten.	Entwickler von Datenbankanwendungen
Verteilte Anwendungen entwickeln (Del5dap.hlp)	Beschreibt die Entwicklung verteilter Anwendungen und enthält Informationen über CORBA, DCOM, MTS, HTTP und Sockets.	Entwickler von Client/Server-Anwendungen
Benutzerdefinierte Komponenten entwickeln (Del5cw.hlp)	Informiert über die Entwicklung benutzerdefinierter Delphi-Komponenten und beschreibt, wie solche Komponenten entworfen, erstellt, getestet und installiert werden.	Komponentenentwickler

Tabelle 1.1 Dateien der Online-Hilfe (Fortsetzung)

Hilfedatei	Inhalt	Zielgruppe
COM-basierte Anwendungen entwickeln (Del5com.hlp)	Beschreibt, wie mittels COM verteilte Anwendungen erstellt werden und enthält Informationen über COM-Objekte, MTS-Komponenten, Automatisierungs-Server und -Controller, ActiveX-Steuerelemente und Typbibliotheken. Außerdem wird erklärt, wie automatisch generierte Typbibliotheken mit dem Typbibliotheks-Editor bearbeitet werden.	Entwickler von Client/Server-Anwendungen
Object-Pascal-Referenz (Del5op.hlp)	Enthält eine formale Definition der Sprache Object Pascal. Behandelt werden Themen wie Datei-E/A, Stringmanipulation, Programmsteuerung, Datentypen und Spracherweiterungen.	Entwickler, die detaillierte Informationen über Object Pascal benötigen
Benutzerdefinierte Hilfe (OpenHelp.hlp)	Erläutert, wie Delphi-Hilfedateien konfiguriert werden. Mit OpenHelp kann jede Windows-Hilfedatei (HLP) aus dem Hilfesystem entfernt oder darin eingefügt werden.	Entwickler, die das Hilfesystem anpassen wollen

Auch für die Zusatzprodukte, die zum Lieferumfang der verschiedenen Delphi-Versionen gehören, gibt es Hilfedateien:

- Integrated Translation Environment (ITE)
- Borland Database Engine (BDE)
- BDE-Administrator
- Datenbank-Explorer
- Local SQL, SQL Builder und SQL Monitor
- Editor für Package-Sammlung
- Help Workshop
- QuickReport
- TeeChart
- InterBase und InterBase Express
- Referenz der CORBA-Komponenten-Bibliothek
- Hilfe für verschiedene Komponenten (FastNet Time, DayTime, Echo, Finger, HTTP, NNTP, POP3, Powersock, SMTP, UDP, URL Encode/Decode, UUprocessor, Stream- und Msg-Komponenten)

Sie finden alle Hilfedateien im Unterverzeichnis HELP des Hauptverzeichnisses von Delphi.

Informationen über die individuelle Anpassung des Hilfesystems finden Sie unter »Das Hilfesystem anpassen« auf Seite 4-11.

Gedruckte Dokumentation

Die vorliegende *Einführung* gibt Ihnen einen ersten Überblick über Delphi. Wie Sie zusätzliche Dokumentation in Buchform bestellen können, erfahren Sie auf der Web Site von Inprise (wwwborland.de).

Inprise Developer Support Services

Um den Bedürfnissen der Delphi-Entwickler zu entsprechen, bietet Inprise eine Reihe zusätzlicher Service-Leistungen an. Informationen über diese Leistungen finden Sie unter der URL
http://www.borland.com/devsupport/delphi.

Auf dieser Web-Site finden Sie auch weitere technische Informationen zu Delphi sowie Antworten auf häufig gestellte Fragen (FAQs).

Von dieser Web-Site aus haben Sie auch Zugang zu zahlreichen Newsgroups, in denen Delphi-Entwickler Informationen und Tips austauschen. Sie finden hier auch eine Liste mit Büchern über Delphi.

Informationen über das Jahr-2000-Problem sowie über unsere Produkte finden Sie unter der URL
http://www.inprise.com/devsupport/y2000/.

Mehr Informationen erhalten Sie von unserem Kundendienst unter 0130 82 08 66

Typographische Konventionen

Zur Kennzeichnung besonderer Textteile werden in diesem Handbuch folgende Schriftarten verwendet.

Tabelle 1.2 Typographische Konventionen

Schriftart	Bedeutung
`Schreibmaschinenschrift`	Schreibmaschinenschrift kennzeichnet Text, der am Bildschirm angezeigt wird, sowie Programmcode. Ferner wird dadurch Text gekennzeichnet, den Sie eingeben müssen.
Fettschrift	Fettgeschriebene Wörter im Text oder in Codebeispielen kennzeichnen reservierte Wörter oder Compiler-Optionen.
Kursivschrift	Kursiv gesetzte Wörter im Text verweisen auf Sprachelemente. Ferner dienen Sie zur Hervorhebung bestimmter Wörter, etwa neuer Fachbegriffe.
Tastennamen	Durch diese Schriftart werden Tasten gekennzeichnet. Beispiel: »Drücken Sie *Esc*, um ein Menü zu verlassen.«

Kapitel

2

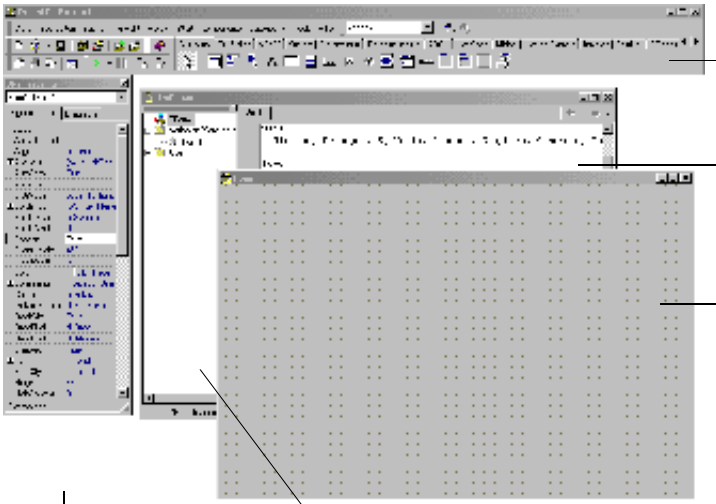
Die Umgebung kennenlernen

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 Windows-Menü *Start* den Befehl *Programme / Delphi.*
- Wählen Sie im Windows-Menü *Start* den Befehl *Ausführen*, und geben Sie dann *Delphi32* ein.
- Doppelklicken Sie im Verzeichnis \BIN von Delphi auf DELPHI32.EXE.

Zu Beginn sehen Sie einige der wichtigsten Tools, die Ihnen in der integrierten Entwicklungsumgebung (IDE) vn Delphi zur Verfügung stehen:



Das Entwicklungsmodell von Delphi basiert auf *Two-way Tools*. Das bedeutet, daß Sie problemlos zwischen visuell orientierten und textbezogenen Programmierwerkzeugen wechseln können. Nachdem Sie beispielsweise mit dem Formular-Designer Schaltflächen und andere grafische Elemente arrangiert haben, können Sie sich in der zugehörigen .DFM-Datei 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.

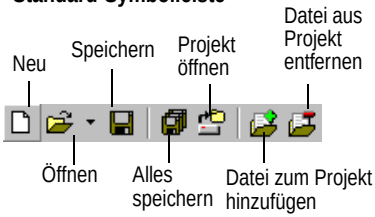
Innerhalb der IDE (Integrated Development Environment) sind alle Programmier-Tools in bequemer Reichweite. Sie können Projekte verwalten, Anwendungen entwickeln, Quelltext schreiben, mit Datenbanken arbeiten, Anwendungen compilieren, testen oder debuggen und Klassenbibliotheken durchsuchen, ohne die IDE zu verlassen.

Einzelheiten über die organisatorische Handhabung sowie die Konfiguration der IDE finden Sie in Kapitel 4, »Die Umgebung anpassen«.

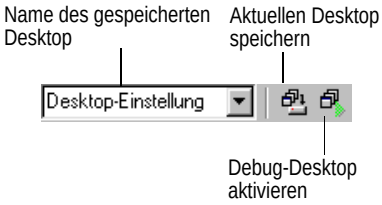
Mit Symbolleisten, Menüs und Tastenkürzeln arbeiten

Die im Hauptfenster von Delphi enthaltenen Symbolleisten ermöglichen einen raschen Zugriff auf die gebräuchlichsten Operationen. Jede dieser Operationen entspricht einem Befehl, der in einem der Drop-down-Menüs enthalten ist.

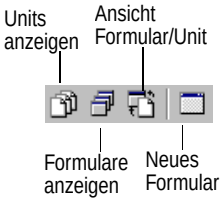
Standard-Symbolleiste



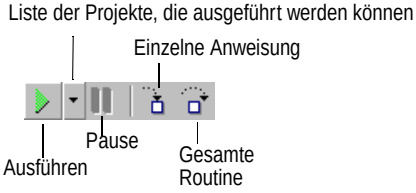
Symbolleiste Desktop



Ansicht-Symbolleiste



Debug-Symbolleiste



Wenn Sie den Mauszeiger auf eine Schaltfläche setzen, wird ihre Funktion und, falls vorhanden, auch das entsprechende Tastenkürzel eingeblendet.

Mit dem lokalen Menü können Sie eine Symbolleiste verbergen. Um eine verborgene Symbolleiste wieder anzuzeigen, markieren Sie sie mit *Ansicht / Symbolleisten*.

Für zahlreiche Operationen stehen sowohl Tastenkürzel als auch Symbolschaltflächen zur Verfügung. Wenn ein Tastenkürzel definiert ist, so wird es im Drop-down-Menü rechts neben dem betreffenden Befehl angezeigt.

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.

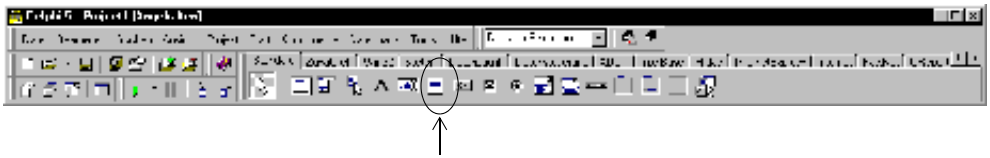
Die Symbolleiste läßt sich an Ihre individuellen Bedürfnisse anpassen. So können Sie beliebige Befehle in sie integrieren oder Teile von ihr an andere Stellen verlagern. Informationen hierzu finden Sie unter »Menüs und Symbolleisten organisieren« auf Seite 4-4.

Mit Hilfe der Desktop-Symbolleiste können Sie unterschiedliche Desktop-Arrangements benennen und speichern.

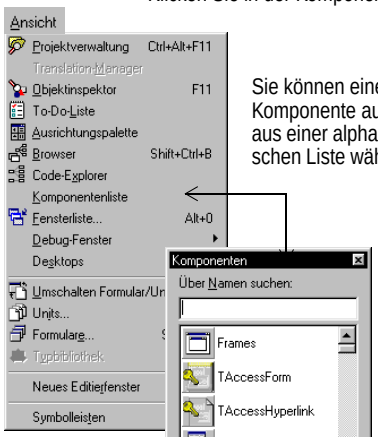
Komponenten in einem Formular plazieren

Wenn Sie die Oberfläche einer Delphi-Anwendung erstellen, plazieren Sie Komponenten in einem Formular, legen ihre Eigenschaften fest und schreiben ihre Ereignisbehandlungsroutinen.

In der Komponentenpalette sind die Komponenten nach ihrer Funktion gruppiert.

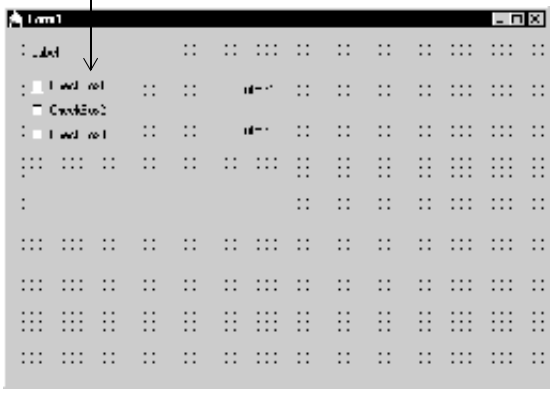


Klicken Sie in der Komponentenpalette auf eine Komponente.

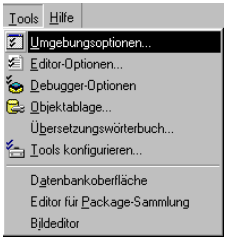


Sie können eine Komponente auch aus einer alphabetischen Liste wählen.

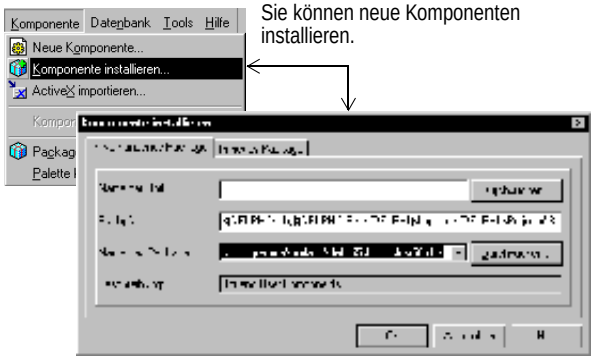
Klicken Sie dann im Formular an der Stelle, an der Sie die Komponente plazieren wollen.



Sie können die Palette anders anordnen und ihr neue Registerkarten hinzufügen. Klicken Sie dazu unter *Tools / Umgebungsoptionen* auf das Register *Palette*.

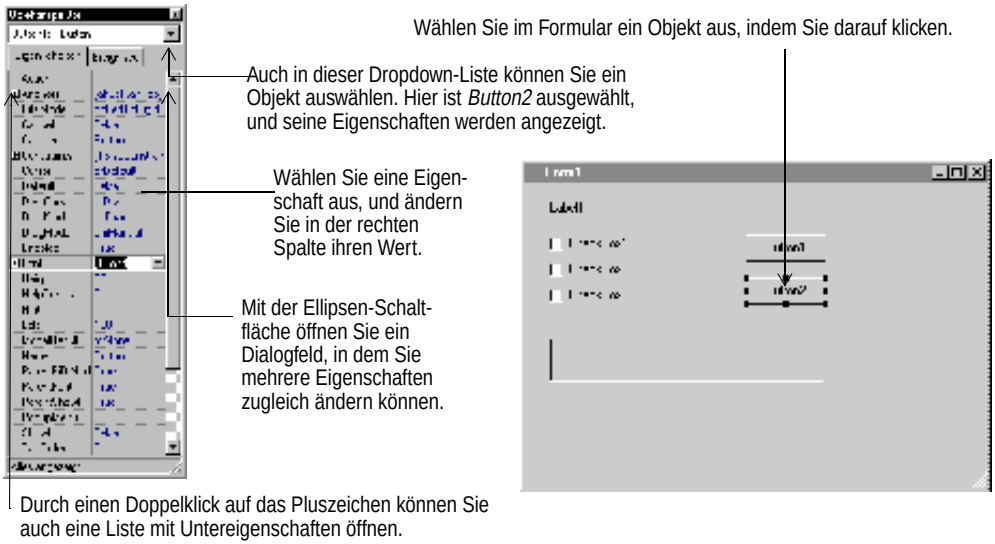


Sie können neue Komponenten installieren.



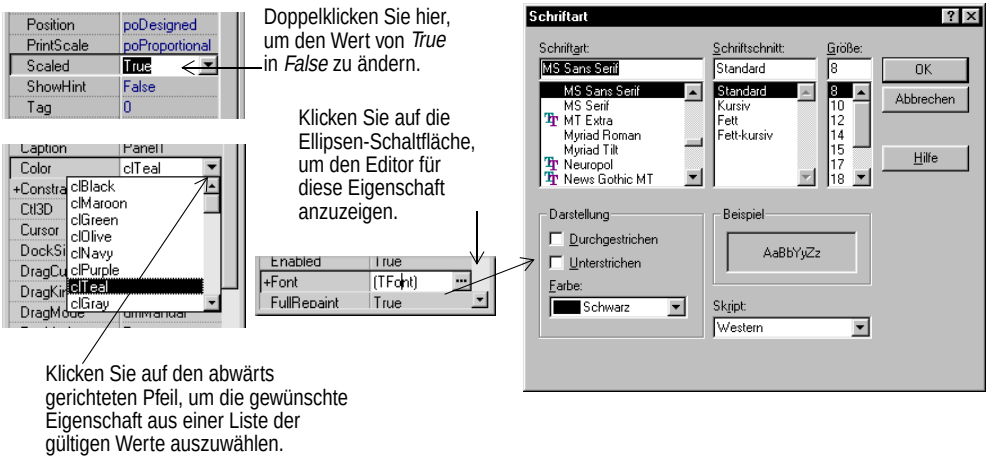
Aussehen und Verhalten von Komponenten ändern

Mit Hilfe des Objektinspektors können Sie das Erscheinungsbild und das Verhalten von Komponenten auf einfache Weise anpassen. Wenn Sie im Formular eine Komponente auswählen, werden im Objektinspektor ihre Eigenschaften und Ereignisse angezeigt.

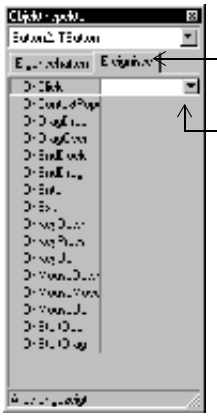


Viele Eigenschaften haben Vorgabewerte, z.B. *True* oder *False*, eine bestimmte Farbe oder einen bestimmten Zahlenwert. Bei Eigenschaften, denen ein Boolescher Wert zugewiesen werden muß (*True* oder *False*), können Sie durch einen Doppelklick auf die rechte Spalte den jeweils anderen Wert aktivieren. Bei Eigenschaften komplexer Art werden die Werte mit Hilfe von Eigenschaftseditoren zugewiesen. In diesen Fäl-

len wird beim Anklicken eines solchen Eigenschaftswertes eine Ellipsen-Schaltfläche (drei Auslassungspunkte) angezeigt.



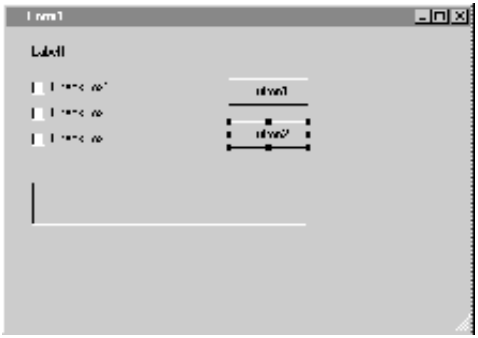
Ereignisse auswählen



Klicken Sie im Objektinspektor auf das Register *Ereignisse* , um die Ereignisse anzuzeigen, auf die eine Komponente reagieren kann. Hier ist die Komponente *Button2* ausgewählt, und ihr Typ *TButton* wird angezeigt.

Wählen Sie ein Ereignis aus der Dropdown-Liste.

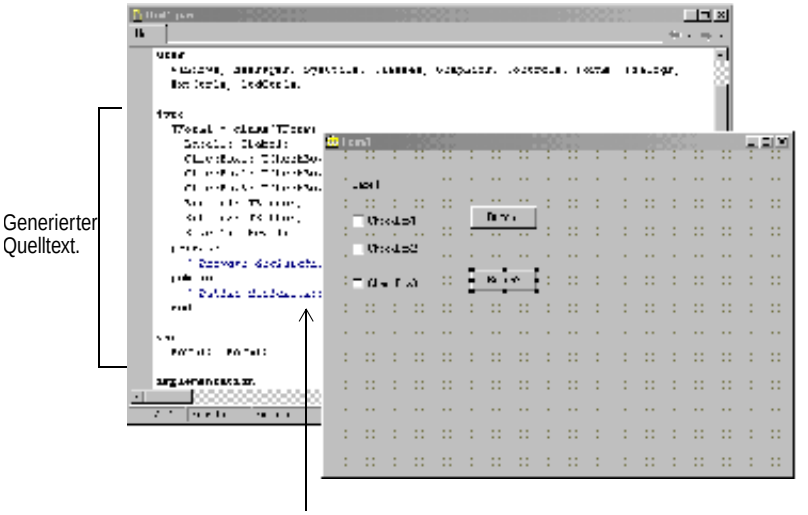
Wenn Sie in der Wertespalte doppelklicken, erzeugt Delphi den Rumpf der Ereignisbehandlungsroutine. Sie brauchen nur noch den Quelltext einzugeben.



Quelltext anzeigen und bearbeiten

Während Sie visuell die Benutzeroberfläche Ihrer Anwendung entwerfen, generiert Delphi den zugrundeliegenden Pascal-Quelltext. Wenn Sie die Eigenschaften von Formularen und Komponenten ändern, werden Ihre Änderungen automatisch in die Quelldateien übernommen.

Mit dem integrierten Quelltexteditor können Sie die Quelltextdateien aber auch direkt bearbeiten. Es handelt sich dabei um einen komplett ausgestatteten ASCII-Editor



Um den Editor individuell anzupassen, wählen Sie *Tools / Editor-Optionen*. Sie können dann beispielsweise Tabulatoren setzen, die Tastenbelegung ändern, andere Farbeinstellungen wählen oder die Einstellungen für automatische Abläufe ändern.

Formulardateien anzeigen

Formulare bilden die sichtbare Oberfläche der meisten Delphi-Projekte. Auf ihnen gestalten Sie die Benutzerschnittstelle einer Anwendung. In der Regel arbeiten Sie bei der Erstellung eines Formulars mit den visuellen Tools von Delphi. Ein solches Formular wird dann in einer Formulardatei gespeichert. Formulardateien (.DFM) beschreiben jede auf dem Formular enthaltene Komponente einschließlich der Werte aller persistenter Eigenschaften.

Um sich eine Formulardatei (.DFM) im Editor anzusehen, wählen Sie im lokalen Menü des Formular-Designers den Befehl *Ansicht als Text*. Um dann wieder zur bildhaften Darstellung des Formulars zurückzukehren, wählen Sie im lokalen Menü den Befehl *Ansicht als Formular*.

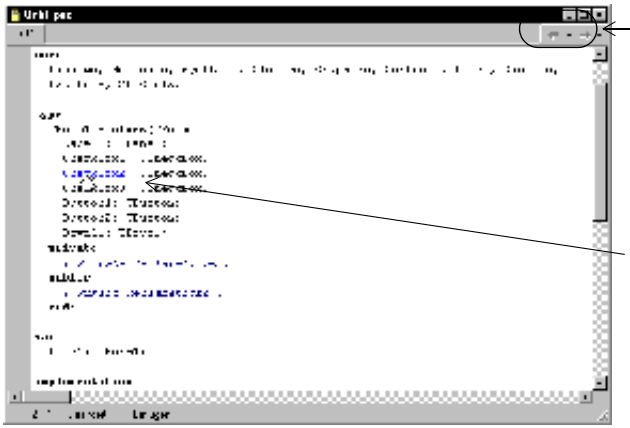
Formulardateien können entweder im Textformat (Vorgabe) oder im Binärformat gespeichert werden. Im Dialogfeld *Umgebungsoptionen* können Sie angeben, welches dieser Formate für neue Formulare verwendet werden soll.

Weitere informationen

Suchen Sie im Hilfeindex nach »Formulardateien«.

Der Editor als Browser

Der Quelltexteditor verfügt über Vor- und Zurück-Schaltflächen, wie Sie sie von einem Web-Browser her kennen. Mit Hilfe dieser Schaltflächen können Sie schnell zu verschiedenen Stellen im Quelltext gelangen. Mit dem nach links gerichteten Pfeil erreichen Sie die Stelle im Quelltext, die sie zuletzt bearbeitet haben. Mit dem nach rechts gerichteten Pfeil landen Sie wieder an der Ausgangsposition.



Verwenden Sie den Editor wie einen Web-Browser.

Drücken Sie *Strg*, und zeigen Sie mit der Maus auf einen beliebigen Bezeichner. Der Mauszeiger nimmt die Form einer Hand an, und der Bezeichner wird unterstrichen und in blauer Schrift dargestellt.

Klicken Sie einmal, um die Definition des Bezeichners anzuzeigen.

Mit dem Links-Pfeil können Sie im Quelltext Stellen aufsuchen, die Sie zuvor bearbeitet haben.

Im Quelltext-Editor können Sie auch durch Drücken von *Strg+Umschalt+*↑ bzw. *Strg+Umschalt+*↓ zwischen der Deklaration einer Prozedur und deren Implementierung wechseln.

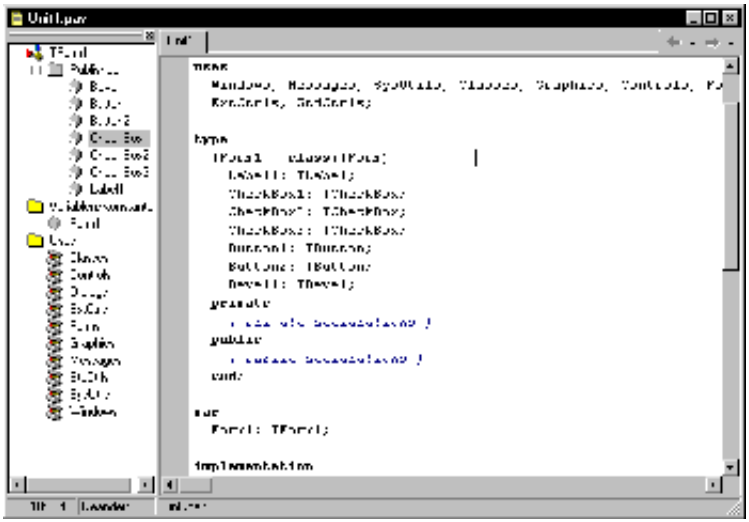
Weitere Informationen

Suchen Sie im Hilfeindex nach »Quelltext-Editor«.

Der Code-Explorer

Wenn im Quelltext-Editor eine Quelldatei geöffnet ist, können Sie sich mit Hilfe des Code-Explorers ein strukturiertes Inhaltsverzeichnis für den Quelltext anzeigen lassen. Der Code-Explorer enthält ein Baumdiagramm, in dem die in Ihrer Unit definierten Typen, Klassen, Eigenschaften, Methoden, globalen Variablen und Routinen dar-

gestellt sind. Außerdem werden die anderen in der **uses**-Klausel aufgeführten Units angezeigt.



Weitere Informationen

Suchen Sie im Hilfeindex nach »Code-Explorer«.

Projektverwaltung

Mit Hilfe der Projektverwaltung behalten Sie stets den Überblick über die Formular- und Unit-Dateien, aus denen sich eine Anwendung zusammensetzt. Um die Projektverwaltung aufzurufen, wählen Sie *Ansicht / Projektverwaltung*.

Mit Hilfe der Projektverwaltung lassen sich die Formular-, Unit-, Ressourcen-, Objekt-, Bibliotheks- und anderen Dateien anzeigen, die zu einem Projekt gehören. Sie können Dateien hinzufügen oder entfernen und mittels Doppelklick jede der angezeigten Dateien öffnen.

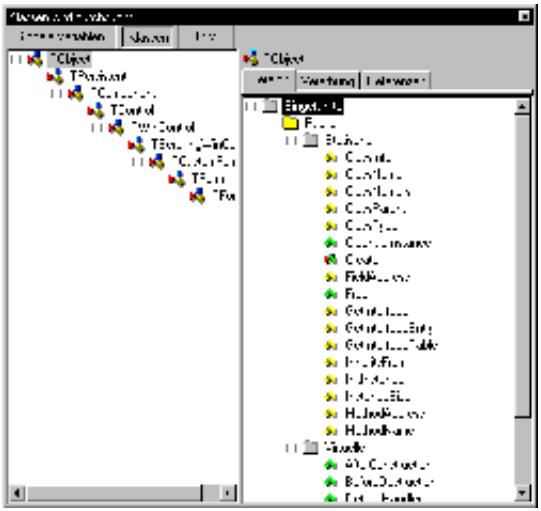
Aufeinander bezogene Projekte lassen sich zu einer einzelnen *Projektgruppe* zusammenfassen. Mit Hilfe einer solchen Projektgruppe lassen sich beispielsweise mehrschichtige Anwendungen verwalten oder DLL-Dateien zusammen mit den darauf zugreifenden EXE-Dateien ablegen.

Weitere Informationen

Suchen Sie im Hilfeindex nach »Projektverwaltung«.

Elemente und Struktur eines Projekts anzeigen

Wie weiter oben bereits erwähnt, läßt sich mit Hilfe des Code-Explorers eine Unit in allen Einzelheiten analysieren. Um einen größeren Überblick über die in einem Projekt enthaltenen Elemente zu bekommen, benutzen Sie den Projekt-Browser. Darin werden die Objekthierarchien, Units und globalen Symbole angezeigt, die zum gesamten Projekt gehören. Um den Projekt-Browser aufzurufen, wählen Sie *Ansicht / Browser*.



Sie können den Anzeigebereich des Projekt-Browsers auch in der Weise ausdehnen, daß er sämtliche Symbole umfaßt, die in der VCL-Objekthierarchie von Delphi zur Verfügung stehen. Wählen Sie dazu *Tools / Umgebungsoptionen*, und aktivieren Sie auf der Seite *Explorer* das Feld *Alle Symbole (mit VCL)*.

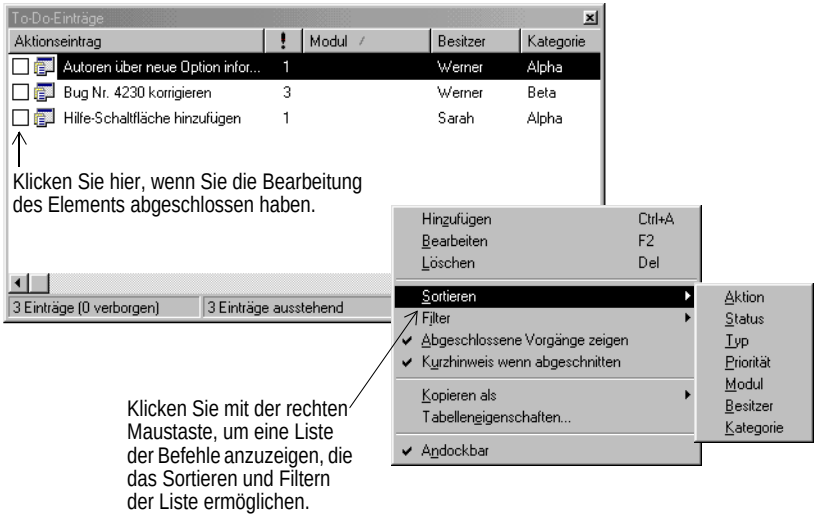
Der Projekt-Browser verfügt über drei Registerkarten, in denen Klassen, Units und globale Symbole aufgeführt sind. Auf der durch *Tools / Umgebungsoptionen* aufgerufenen *Explorer*-Seite können Sie den Anzeigebereich des Projekt-Browser festlegen und die Gruppierung der angezeigten Elemente steuern.

Weitere Informationen

Suchen Sie im Hilfeindex nach »Projekt-Browser«.

To-DoListen erstellen

In To-Do-Listen werden Elemente erfaßt, die für die Fertigstellung eines Projekts benötigt werden. Sie können projektbezogene Elemente direkt in eine solche Liste aufnehmen, oder Sie können bestimmte Elemente unmittelbar in den Quellcode einfügen. Um projektspezifische Informationen hinzufügen oder anzuzeigen, wählen Sie *Ansicht / To-Do-Liste*.



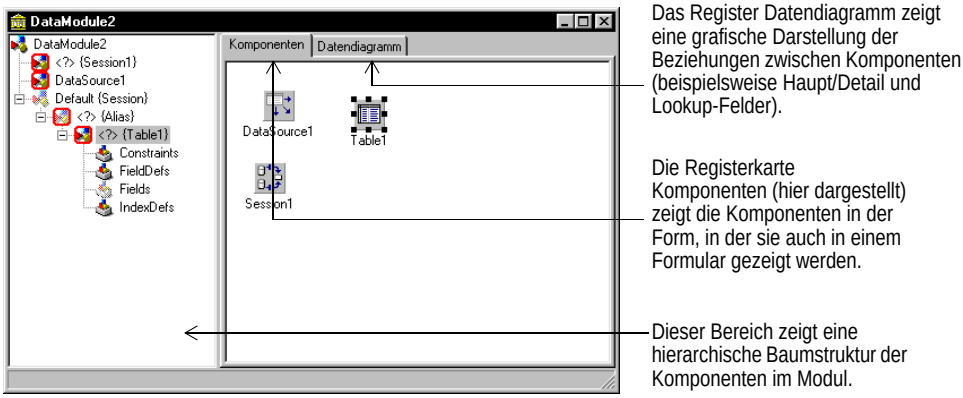
Weitere Informationen

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

Datenmodule entwerfen

Ein Datenmodul ist ein besonderes Formular mit nichtvisuellen Komponenten. Solche Komponenten *können* zusammen mit visuellen Steuerelementen in ein gewöhnliches Formular eingefügt werden. Wenn Sie jedoch die Absicht haben, Gruppen von Datenbank- und Systempbjekten wiederzuverwenden, oder wenn Sie diejenigen Teile Ihrer Anwendung isolieren möchten, die Datenbank-Connectivity und Business rules betreffen, lassen sich Datenmodule als zweckmäßige Organisationswerkzeuge einsetzen.

Mit Hilfe des Datenmodul-Designers ist die Erstellung von Datenmodulen sehr einfach. Wählen Sie dazu *Datei / Neu*, und klicken Sie zweimal auf *Datenmodul*.



Delphi öffnet im Datenmodul-Designer ein leeres Datenmodul, zeigt im Quelltexteditor die Unit-Datei für das neue Modul an und fügt das Modul zum aktuellen Projekt hinzu. Wenn Sie ein vorhandenes Datenmodul erneut öffnen, werden im Datenmodul-Designer seine Komponenten angezeigt.

Weitere Informationen

Suchen Sie im Hilfeindex nach »Datenmodul-Designer« oder nach »Datenmodule«.

Projekt- und Umgebungsoptionen einstellen

Im Dialogfeld *Projektoptionen*, auf das Sie mit *Projekt / Optionen* zugreifen, werden Einstellungen für das Compilieren und Linken vorgenommen, bestimmte Suchpfade und Ausgabeverzeichnisse angegeben, Informationen über die Projektversion hinterlegt und andere Festlegungen getroffen, die eine einzelne Anwendung betreffen. Änderungen an diesen Einstellungen gelten immer nur für das aktuelle Projekt; wenn jedoch das Kontrollfeld *Vorgabe* aktiviert ist, werden Ihre Einstellungen zugleich als Standardeinstellungnen für neue Projekte gespeichert. (Siehe »Standard-Projektoptionen festlegen« auf Seite 4-5.)

Im Dialogfeld *Umgebungsoptionen*, auf das Sie mit *Tools / Umgebungsoptionen* zugreifen, werden die globalen IDE-Einstellungen für alle Projekte festgelegt. Dies umfaßt sowohl Einstellungen, die das Aussehen und das Verhalten der IDE betreffen, als auch bestimmte Suchpfade und Ausgabeverzeichnisse. Informationen über einige dieser Einstellungen finden Sie im Abschnitt »Einstellungen für Tools festlegen« auf Seite 4-7.

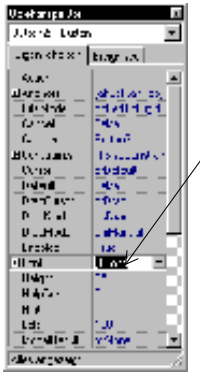
Weitere Informationen

Wenn Sie detaillierte Informationen über die Einstellungen benötigen, die im Dialogfeld *Projektoptionen* bzw. *Umgebungsoptionen* festgelegt werden, klicken Sie auf der betreffenden Registerkarte die Schaltfläche *Hilfe* an. Oder Sie suchen im Hilfeindex

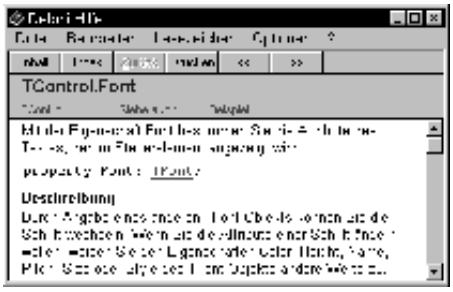
nach den Stichwörtern »Projektoptionen (Dialogfeld)« bzw. »Umgebungsoptionen (Dialogfeld)«.

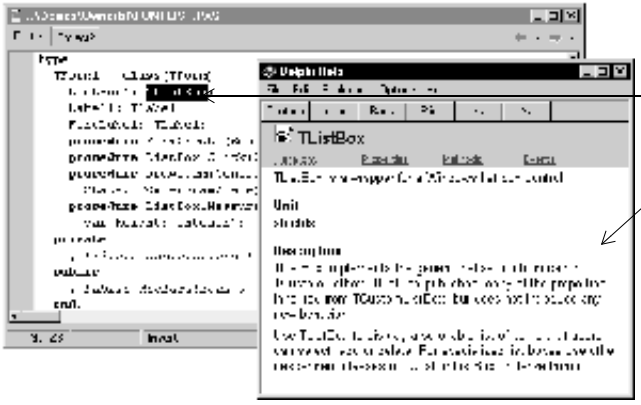
Hilfe

Die Online-Hilfe dokumentiert umfassend die VCL und andere Teile von Delphi. Sie können auf viele verschiedene Arten auf das Hilfesystem zugreifen. Im folgenden sind einige dieser Möglichkeiten aufgeführt:

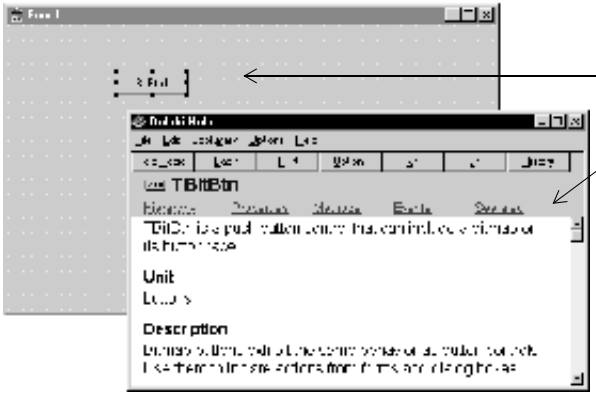


Um die VCL-Hilfe aufzurufen, markieren Sie im Objektinspektor den Namen einer Eigenschaft oder eines Ereignisses und drücken *F1*.



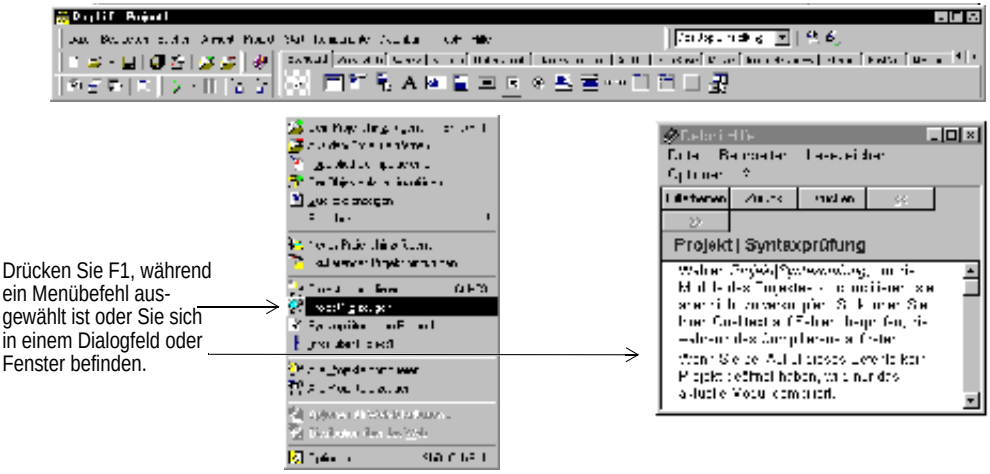


Markieren Sie im Quelltexteditor ein Schlüsselwort oder ein VCL-Element, und drücken Sie *F1*.



Markieren Sie im Formular-Designer ein Objekt, und drücken Sie *F1*.

Die Hilfe umfaß jeden beliebigen Teil der Entwicklungsumgebung, also auch Menübefehle, Dialogfelder, Fenster, Symbolleisten und Komponenten.



Durch Drücken der Schalfläche *Hilfe* in einem beliebigen Dialogfeld läßt sich die kontextsensitive Online-Dokumentation aufrufen.

Fehlermeldungen vom Compiler und Linker werden in einem besonderen Fenster unterhalb des Quelltexteditors angezeigt. Um Hilfe zu einem Compilierfehler zu erhalten, markieren Sie in der Liste die betreffende Meldung und drücken *F1*.

Hilfe beim Programmieren

Beim Programmieren stehen verschiedene Hilfe-Tools zur Verfügung. Diese Tools zeigen im Quelltexteditor kontextsensitive Popup-Fenster an.

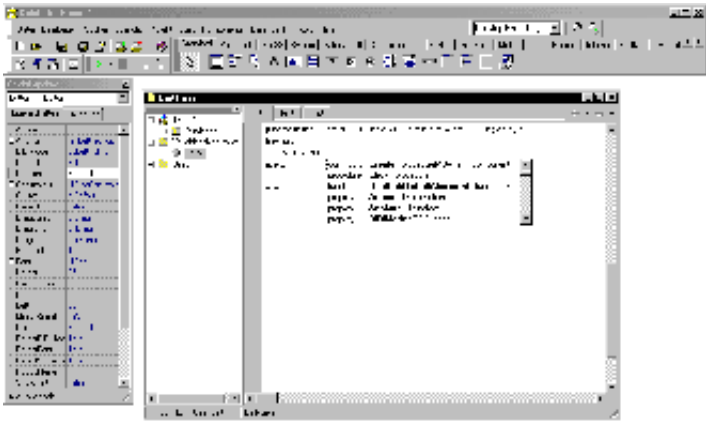
Tabelle 2.1 Programmierhilfen

Tool	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. Wenn Sie den Anfang eines Zuweisungsstatements eingeben und *Strg+Leertaste* drücken, wird eine Liste mit den für diese Variable gültigen Werten angezeigt. 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.
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.

Tabelle 2.1 Programmierhilfen (Fortsetzung)

Tool	Funktionsweise
Auswertung von Ausdrücken	Wenn während einer Debugger-Sitzung das Programm angehalten wurde, können Sie den Mauszeiger auf eine beliebige Variable setzen, um sich deren aktuellen Wert anzeigen zu lassen.
Symbolinformationen	Wenn Sie Ihren Quelltext bearbeiten, können Sie den Maus-zeiger auf einen beliebigen Bezeichner setzen, um sich dessen Deklaration anzeigen zu lassen.

Um diese Tools zu konfigurieren, wählen Sie *Tools / Umgebungsoptionen* und öffnen die Registerkarte *Editor-Optionen*.



Wenn Sie `Button1.` eingeben, erscheint eine Liste mit den Eigenschaften, Methoden und Ereignissen dieser Klasse.

Um einen Listeneintrag in den Quelltext einzufügen, markieren Sie ihn und drücken *Return*.

Vervollständigung von Klassen

Mittels der Vervollständigung von Klassen läßt sich Rumpfcode für Klassen generieren. Dazu setzen Sie den Cursor an eine beliebige Stelle innerhalb einer Klassendeklaration und drücken entweder *Strg+Umschalt+C* oder führen einen Rechtsklick aus und wählen aus dem lokalen Menü den Befehl *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 konfigurieren, wählen Sie *Tools / Umgebungsoptionen* und klicken auf die Registerkarte *Explorer*.

Weitere Informationen

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

Fehlersuche in Anwendungen

Die IDE verfügt über einen integrierten Debugger, mit dem Sie Fehler im Quelltext lokalisieren und beseitigen können. Der Debugger gibt Ihnen die Möglichkeit, die Programmausführung zu steuern, Variablen zu überwachen und während des Programmablaufs Datenwerte zu ändern. Sie können den Code Zeile für Zeile ausführen lassen und so den Zustand des Programms an jedem Haltepunkt überprüfen.

Die Debugger-Befehle finden Sie im Menü *Start* und in der Symbolleiste.

Debugger-Schaltflächen

Die Schaltfläche *Start*

Damit Sie mit dem Debugger arbeiten können, muß Ihr Programm mit Debug-Informationen compiliert worden sein. Wählen Sie dazu *Projekt / Optionen*, öffnen Sie dann die Registerkarte *Compiler*, und aktivieren Sie das Kontrollkästchen *Debug-Informationen*. Anschließend können Sie eine Debugger-Sitzung starten, indem Sie Ihr Programm in der IDE ausführen. Um den Debugger zu konfigurieren, wählen Sie *Tools / Debugger-Optionen*.

Zu den zahlreichen Fenstern des Debuggers gehören die Fenster *Haltepunkte*, *Aufruf-Stack*, *Überwachte Ausdrücke*, *Lokale Variablen*, *Threads*, *Module*, *CPU* und *Ereignisprotokoll*. Sie finden diese Fenster unter *Ansicht / Debug-Fenster*. Zur bequemeren Handhabung lassen sich verschiedene Debugger-Fenster aneinander andocken. Informationen dazu finden Sie im Abschnitt »Tool-Fenster andocken« auf Seite 4-1.

Bei der Fehlersuche steht Ihnen eine Vielzahl von Informationen zur Verfügung. Wählen Sie die gewünschten Fenster unter *Ansicht / Debug-Fenster*.

Viele der Debugger-Ansichten können aneinander angedockt werden. Klicken Sie auf die Register, um die verschiedenen Ansichten anzuzeigen.

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. Einzelheiten hierzu finden Sie unter »Desktop-Einstellungen anpassen« auf Seite 4-5.

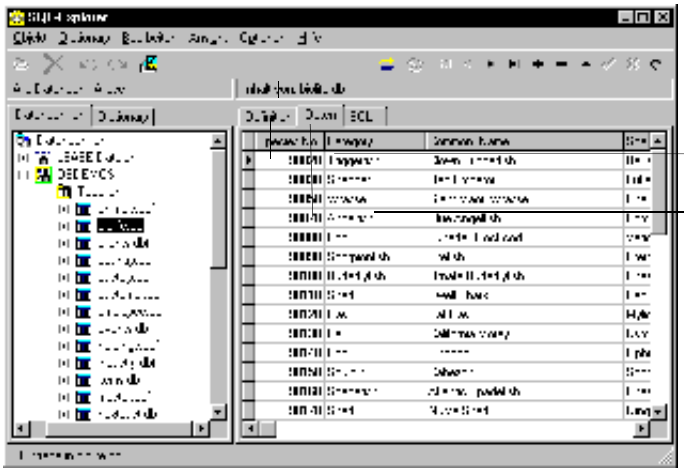
Bei verteilten Anwendungen unterstützen einige Delphi-Versionen auch Remote- und Multiprozeß-Debugging sowohl vom Client als auch vom Server aus. Um auf Remote-Debugging umzuschalten, wählen Sie *Start / Parameter*, öffnen die Registerkarte *Extern*. Anschließend wählen Sie *Projekt / Optionen*, öffnen die Registerkarte *Linker* und aktivieren *Mit externen Debug-Symbolen*.

Weitere Informationen

Suchen Sie im Hilfeindex nach »Debuggen«.

Datenbanken abfragen

Der SQL-Explorer (in einigen Delphi-Editionen wird er als Datenbank-Explorer bezeichnet) gibt Ihnen die Möglichkeit, während der Anwendungsentwicklung direkt mit einem Remote-Datenbankserver zu arbeiten. So können Sie beispielsweise Dateneingabe-Beschränkungen importieren oder Tabellen erstellen, löschen und umstrukturieren, während Sie eine Datenbankanwendung entwickeln.



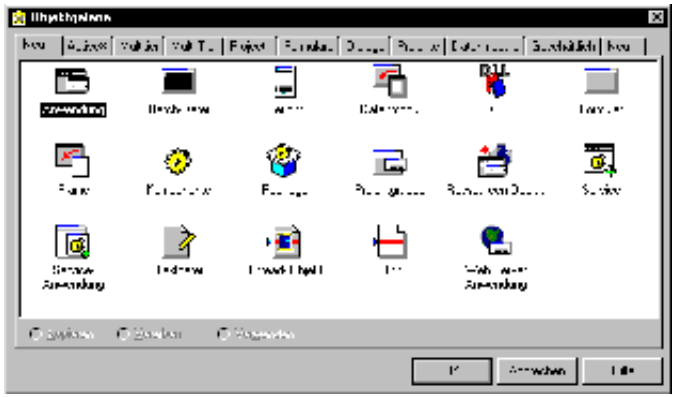
Weitere Informationen

Rufen Sie mit *Datenbank / Explorer* den Explorer auf, und drücken Sie F1. Oder suchen Sie im Hauptindex des Hilfesystems nach dem Stichwort »Datenbank-Explorer«.

Objekte als Vorlagen speichern

Die Objektablage enthält Formulare, Dialogfelder, Datenmodule, Experten, DLLs, Beispielanwendungen und andere Objekte, die die Anwendungsentwicklung vereinfachen. Wenn Sie ein neues Projekt beginnen, wählen Sie *Datei / Neu*, um das Dialogfeld *Objektgalerie* zu öffnen, und sehen Sie dann die Objektablage durch. Möglicherweise enthält sie bereits ein Objekt, das dem von Ihnen benötigten Objekt ähnlich ist, so daß Sie darauf aufbauen können..

Die Objektablage enthält viele Registerkarten mit Objekten wie Formularen, Rahmen, Units und Batch-Dateien sowie Experten zum Erstellen spezieller Komponenten.



Sie können ein vorhandenes Objekt kopieren, erben oder referenzieren.

Wenn Sie selbstentworfene Objekte in die Objektablage aufnehmen, können Sie sie jederzeit wiederverwenden oder auch anderen Entwicklern zur Verfügung stellen. Auf diese Weise lassen sich ganze Familien von Anwendungen mit einheitlicher Oberfläche und Funktionalität erstellen. Dieses Vorgehen spart Entwicklungszeit und sorgt für eine gleichbleibend hohe Qualität. Außerdem dient die Objektablage als zentraler Aufbewahrungsort für Tools, auf die alle Mitglieder eines Entwicklerteams über ein Netzwerk zugreifen können.

Um ein Objekt in die Objektablage aufzunehmen, klicken Sie mit der rechten Maustaste das Dialogfeld *Objektgalerie* an und wählen dann den Befehl *Eigenschaften*, oder aber Sie wählen aus dem Hauptmenü *Tools / Objektablage*.

Weitere Informationen

Suchen Sie im Hilfeindex das Stichwort »Objektablage«. Sie können sich aber auch durch die Auswahl von *Datei / Neu* selbst einen Überblick verschaffen und nachsehen, welche Vorlagen und Experten für Ihre Anwendungen brauchbar sind. Welche Objekte Ihnen als Ausgangsmaterial zur Verfügung stehen, hängt davon ab, mit welcher Delphi-Version Sie arbeiten.

Kapitel

3

Die erste Anwendung – eine kurze Einführung

Am besten machen Sie sich mit Delphi vertraut, indem Sie eine kleine Anwendung schreiben. Im folgenden erstellen Sie ein Programm, mit dem Sie die Daten der mitgelieferten Meeresfauna-Beispieltabelle anzeigen können. Dazu richten Sie zuerst den Zugriff auf die Tabelle ein. Anschließend erstellen Sie eine Ereignisbehandlungsroutine, in der das Standarddialogfeld *Speichern unter* geöffnet wird. Dadurch können Informationen aus der Datenbanktabelle in eine Datei geschrieben werden.

Hinweis Sie können dieses Tutorial nur nachvollziehen, wenn Sie mit der Professional- oder der Enterprise-Version von Delphi arbeiten. Hierbei werden Datenbankzugriffs-Features eingesetzt, die in der Standardversion von Delphi nicht zur Verfügung stehen.

Eine neue Anwendung beginnen

Erstellen Sie als erstes ein Verzeichnis für die Quelltextdateien Ihrer neuen Anwendung.

1. Erstellen Sie im Verzeichnis PROJECTS des Delphi-Hauptverzeichnisses das Unterverzeichnis FAUNA.
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 / Neue Anwendung* ein neues Projekt erstellen.

   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 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.

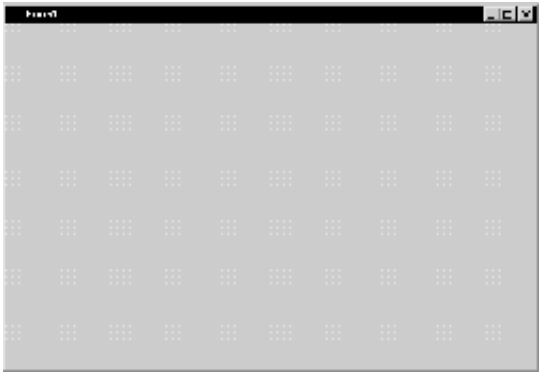
Jedes Formular hat seine eigenen Unit- und Formulardateien.

3 Wählen Sie *Datei / Alles speichern*. Wenn das Dialogfeld *Speichern* erscheint, wechseln Sie in das Verzeichnis FAUNA und speichern jede Datei unter dem vorgegeben Namen.

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

Sie werden jetzt vielleicht feststellen, daß beim Speichern neben den obengenannten Dateien noch weitere Dateien im Projektverzeichnis erstellt wurden. Machen Sie sich darüber keine Gedanken. Sie sollten aber diese zusätzlichen Dateien auf keinen Fall löschen.

Wenn Sie ein neues Projekt öffnen, wird eine grafische Darstellung des Projektformulars mit dem Standardnamen *Form1* angezeigt. Sie werden nun die Benutzerschnittstelle und andere Teile der Anwendung erstellen, indem Sie in dieses Formular Komponenten einfügen.

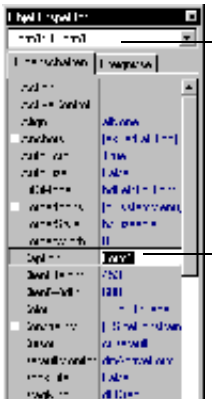


Das neue Standardformular verfügt über Schaltflächen zum Maximieren und Minimieren, ein Schließfeld und ein Systemmenü.

Wenn Sie die Anwendung jetzt mit *F9* starten, werden Sie feststellen, daß alle Schaltflächen bereits funktionieren.

Um in den Entwurfsmodus zurückzukehren, klicken Sie auf das Schließfeld (**X**) des Formulars.

Neben dem Formular wird der Objektinspektor angezeigt, in dem Sie für die eingefügten Komponenten Eigenschaftswerte festlegen können.



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

Die Eigenschaften des ausgewählten Formulars werden im Objektinspektor angezeigt.

Eigenschaftswerte festlegen

Wenn Sie zur Festlegung von Eigenschaftswerten den Objektinspektor benutzen, können Sie Delphi die Wartung des Quellcodes überlassen. Die auf diese Art festgelegten Eigenschaftswerte werden als *Einstellungen zur Entwurfszeit* bezeichnet.

- Setzen Sie die Hintergrundfarbe (Eigenschaft *Color*) von *Form1* auf Aqua.

  Um den Wert der Formulareigenschaft *Color* zu ändern, klicken Sie im Objektinspektor auf die Dropdown-Liste, die rechts neben dem Namen der Eigenschaft angezeigt wird. Wählen Sie anschließend aus der Liste mit den vordefinierten Farbwerten den Eintrag *clAqua*.

Objekte zum Formular hinzufügen

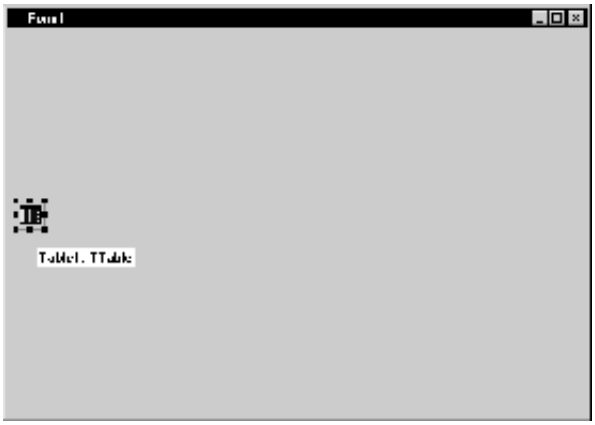
Die Komponentenpalette enthält eine Vielzahl von Komponenten, die Sie in Ihren Anwendungen verwenden können. Damit Sie jederzeit schnell darauf zugreifen können, sind die Komponenten funktionell auf verschiedene Registerkarten verteilt. Jede Komponente wird durch ein bestimmtes Symbol repräsentiert. Um eine Komponente in ein Formular einzufügen, wählen Sie sie in der Komponentenpalette aus und klikken im Formular auf die Stelle, an der sie eingefügt werden soll. Sie können auch auf eine Komponente doppelklicken, um sie in der Mitte des Formulars zu plazieren.

Fügen Sie nun ein *Table*- und ein *StatusBar*-Objekt in das Formular ein:

1 Plazieren Sie ein *Table*-Objekt im Formular.

Klicken Sie auf das Register *Datenzugriff* der Komponentenpalette. Um die *Table*-Komponente zu finden, zeigen Sie mit dem Mauszeiger auf eine Komponente. Nach kurzer Zeit wird ein Hinweis mit dem Namen der betreffenden Komponente eingeblendet.

Klicken Sie zuerst auf die Komponente, um sie auszuwählen, und anschließend im Formular, um die Komponente dort einzufügen. Da die *Table*-Komponente ein nichtvisuelles Objekt ist, spielt die Position, an der sie eingefügt wird, keine Rolle. Das Objekt erhält automatisch den Standardnamen *Table1*. (Wenn Sie in einem Formular auf ein Objekt zeigen, werden dessen Name (hier *Table1*) sowie dessen Typ (*TTable*) angezeigt.)

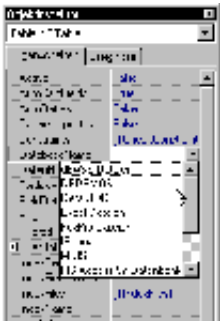


Nichtvisuelle Komponenten werden als Symbole angezeigt.

Jede Delphi-Komponente ist eine *Klasse*. Durch das Einfügen einer Komponente in ein Formular wird eine *Instanz* dieses Objekts erstellt. Sobald sich die Komponente im Formular befindet, generiert Delphi den gesamten Code, der für dieses Objekts erforderlich ist.

2 Setzen Sie die Eigenschaft *DatabaseName* von *Table1* auf *DBDEMOS*. (Dies ist der Alias der Beispieldatenbank, mit der Sie gleich arbeiten werden.)

Dazu markieren Sie auf dem Formular *Table1* und wählen im Objektinspektor die Eigenschaft *DatabasName*. Wählen Sie dann aus der Dropdown-Liste *DBDEMOS*.

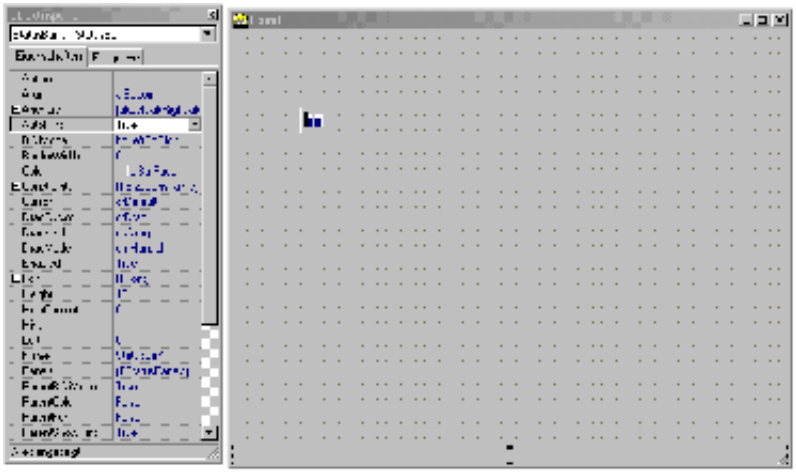


Klicken Sie auf den Abwärtspfeil, um die Dropdown-Liste zu öffnen.

Wählen Sie *DBDEMOS*

3 Doppelklicken Sie in der Registerkarte *Win32* der Komponentenpalette auf die *StatusBar*-Komponente. Dadurch wird am unteren Rand des Formulars eine Statusleiste eingefügt.

4 Setzen Sie die Eigenschaft *AutoHint* der Statusleiste auf *True*. Am besten doppelklicken Sie dazu rechts neben *AutoHint* auf den Wert *False*. (Wenn diese Eigenschaft *True* ist, werden bei der späteren Ausführung des Programms in der Statusleiste Hilfehinweise angezeigt.)

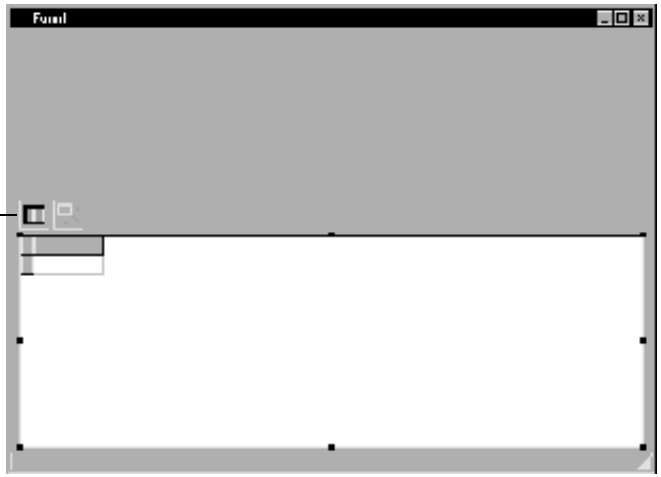


Auf eine Datenbank zugreifen

Als nächstes können Sie die datensensitiven Steuerelemente über eine Datenquelle mit der Datenbank verbinden.

1. Plazieren Sie im Formular eine *DataSource*-Komponente aus der Registerkarte *Datenzugriff* der Komponentenpalette. Da es sich um eine nichtvisuelle Komponente handelt, spielt die Position, an der sie eingefügt wird, keine Rolle. Setzen Sie die Eigenschaft *DataSet* der Komponente auf *Table1*.
2. Klicken Sie auf die Registerkarte *Datensteuerung*, und fügen Sie eine *DBGrid*-Komponente hinzu. Plazieren Sie die Komponente oberhalb der Statusleiste in der linken unteren Ecke des Formulars, und vergrößern Sie sie durch Ziehen der rechten unteren Ecke.

   Bei Bedarf können Sie auch das Formular durch Ziehen der rechten unteren Ecke vergrößern. Es sollte jetzt der nachfolgenden Abbildung entsprechen.

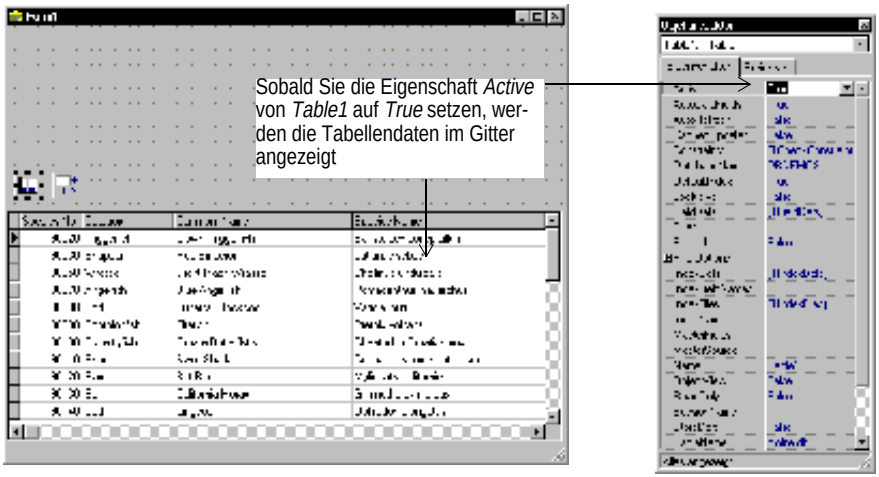


3. Um nun das Datengitter im Formular auszurichten, sind die entsprechenden Eigenschaften von *DBGrid* festzulegen. Dazu klicken Sie im Objektinspektor *Anchors* doppelt an, um die Eigenschaften *akLeft*, *akTop*, *akRight* und *akBottom* anzuzeigen. Setzen Sie dann alle auf *True*.
4. Setzen Sie die Eigenschaft *DataSource* von *DBGrid* auf *DataSource1*. (Dies ist der Standardname der eben in das Formular eingefügten *DataSource*-Komponente.)

   Das Objekt *Table1*, das Sie zuvor im Formular plaziert haben, kann nun fertiggestellt werden.
5. Klicken Sie auf das Objekt *Table1*, um es zu aktivieren, und setzen Sie seine Eigenschaft *TableName* auf *BIOLIFE.DB*. (*Name* ist immer noch *Table1*.) Als nächstes setzen Sie die Eigenschaft *Active* auf *True*.

   Wenn Sie der Eigenschaft *Active* den Wert *True* zuweisen, wird das Gitter mit den Daten der Tabelle BIOLIFE.DB gefüllt. Entspricht das Gitter nicht der folgenden Abbildung, vergewissern Sie sich, daß Sie die Eigenschaften der plazierten Objekte in den vorangegangenen Anweisungen richtig festgelegt haben. (Vergewissern

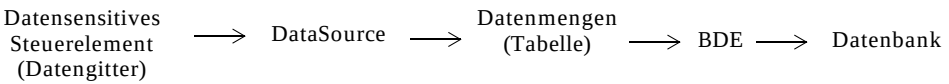
Sie sich auch, daß Sie bei der Installation von Delphi die Dateien der Beispieldatenbank in das Verzeichnis ...\Borland Shared\Data kopiert haben.)



Da die *DBGrid*-Komponente *datensensitiv* ist, zeigt sie die Tabellendaten bereits zur Entwurfszeit an. Sie können dadurch sofort sehen, ob die Verbindung mit der Datenbank korrekt funktioniert. Datensensitiv bedeutet aber nicht, daß die Daten zur Entwurfszeit geändert werden können. Dies ist nur zur Laufzeit mit Hilfe der fertigen Anwendung möglich.

6 Drücken Sie *F9*, um das Projekt zu compilieren und auszuführen. (Sie können das Projekt auch starten, indem Sie entweder in der *Debug*-Symbolleiste das *Start*-Symbol anklicken oder aus dem *Start*-Menü den Befehl *Start* wählen.)

Bei der Verknüpfung der Anwendung mit einer Datenbank wurden drei Komponenten verwendet und verschiedenartige Umwege beschritten. Ein datensensitives Steuerelement (hier eine *DBGrid*-Komponente) zeigt auf ein *DataSource*-Objekt, das wiederum auf ein Datenmengenobjekt (hier ein *Table*-Objekt) zeigt. Das Datenmengenobjekt (*Table1*) schließlich zeigt auf eine tatsächliche Datenbanktabelle (BIOLIFE), auf die über den BDE-Alias *DBDEMOS* zugegriffen wird. (Die Konfiguration von BDE-Aliasen erfolgt mit Hilfe des BDE-Administrators.)



Diese Architektur sieht auf den ersten Blick kompliziert aus, vereinfacht aber auf längere Sicht die Entwicklung und Pflege einer Anwendung. Weitere Informationen hierzu finden Sie in Teil II »Datenbankanwendungen entwickeln« im *Entwicklerhandbuch* oder in der Online-Hilfe.

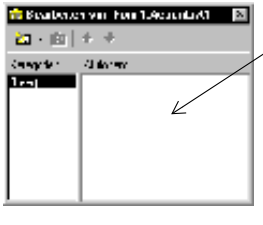
Symbolleisten- und Menüunterstützung hinzufügen

Wenn Sie die Anwendung ausführen, wird sie in einem Fenster geöffnet, das genau Ihrem Formularentwurf entspricht. Die Anwendung ist ein vollständiges Windows-Programm mit einem Systemmenü und Schaltflächen zum Minimieren, Maximieren und Schließen des Fensters. Im Datengitter wird der Inhalt der Tabelle BIOLIFE.DB angezeigt. Mit Hilfe der Bildlaufleisten können Sie auch Daten anzeigen, die außerhalb des aktuellen Bildausschnitts liegen.

Auch wenn Ihre Anwendung bereits die wichtigsten Elemente enthält, so fehlen doch noch einige typische Windows-Features. So verfügen die meisten Windows-Anwendungen aus Gründen der leichteren Bedienbarkeit über Symbolleisten und Menüs.

In diesem Abschnitt werden Sie Ihre Anwendung für zusätzliche grafische Schnittstellenelemente vorbereiten, indem Sie eine *ActionList*-Komponente einrichten. Obwohl Sie Menüs, Symbolleisten und Schaltflächen auch ohne Aktionslisten erstellen können, vereinfachen sie doch die Entwicklung und Pflege einer Anwendung, da sich mit ihrer Hilfe die Antworten auf die Befehle des Benutzers zentralisieren lassen.

1. Klicken Sie auf das **X** in der rechten oberen Ecke, um die Anwendung zu schließen und zum Entwurfsfenster zurückzukehren.
2. Öffnen Sie die Registerkarte *Win32* der Komponentenpalette, und plazieren Sie eine *ImageList*-Komponente im Formular. Die Position spielt keine Rolle, da auch diese Komponente ein nichtvisuelles Objekt ist. Im *ImageList*-Objekt werden die in der Anwendung verwendeten grafischen Symbole für Standardaktionen wie *Ausschneiden* und *Einfügen* gespeichert.
3. Öffnen Sie die Registerkarte *Standard* der Komponentenpalette, und fügen Sie dem Formular eine *ActionList*-Komponente hinzu. Auch dies ist eine nichtvisuelle Komponente.
4. Weisen Sie der Eigenschaft *Images* der Aktionsliste die Komponente *ImageList1* zu.
5. Doppelklicken Sie auf die Aktionsliste, um den Listeneditor zu öffnen.

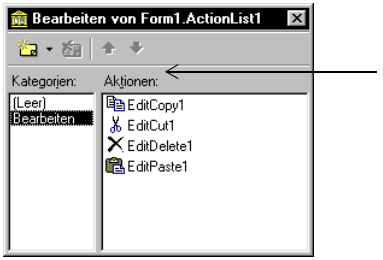


Klicken Sie im Editor für Aktionslisten mit der rechten Maustaste, und wählen Sie *Neue Standardaktion*.

Wählen Sie die gewünschten Aktionen aus, und bestätigen Sie anschließend mit *OK*. Um mehrere Aktionen zugleich auszuwählen, klicken Sie bei gedrückter *Strg*- bzw. *Umschalttaste* auf die betreffenden Einträge.

6. Klicken Sie im Editor für Aktionslisten mit der rechten Maustaste, und wählen Sie *Neue Standardaktion*. Dadurch wird das Dialogfeld *Standardaktionen* geöffnet.

7 Wählen Sie folgende Aktionen aus: *TDataSetFirst*, *TDataSetLast*, *TDataSetNext*, *TDataSetPrior*, *TEditCopy*, *TEditCut* und *TEditPaste* (Verwenden Sie hierbei zur Mehrfachauswahl die *Strg*-Taste). Klicken Sie anschließend auf *OK*.



Sie haben jetzt vordefinierte Standardaktionen mit den entsprechenden Standardsymbolen hinzugefügt.

Diese Elemente werden für eine Symbolleiste und ein Menü verwendet.

8 Klicken Sie auf das Schließfeld (**X**), um den Editor für Aktionslisten zu schließen.

Sie haben jetzt mehrere Standardaktionen hinzugefügt und können mit der Erstellung des Menüs und der Symbolleiste beginnen.

Ein Menü hinzufügen

In diesem Abschnitt erstellen Sie ein Hauptmenü mit drei Elementen (*Datei*, *Bearbeiten* und *Datensatz*). Diesen Elementen werden Sie mit Hilfe der Standardaktionen in der Aktionsliste verschiedene Menüeinträge hinzufügen.

1 Fügen Sie eine *MainMenu*-Komponente aus der Registerkarte *Standard* der Komponentenpalette in das Formular ein. Auch bei dieser nichtvisuellen Komponente ist die Position ohne Bedeutung.

2 Setzen Sie die Eigenschaft *Images* der Komponente auf *ImageList1*.

3 Doppelklicken Sie auf die Komponente, um den Menü-Designer zu öffnen.

4 Geben Sie für die Eigenschaft *Caption* des ersten Hauptmenüeintrags den Wert `&Datei` ein, und drücken Sie anschließend *Return*.

Wenn Sie *&Datei* eingeben und *Return* drücken, wird der betreffende Hauptmenüeintrag angezeigt, und Sie können die erste Menüoption einfügen.

5 Geben Sie `&Speichern` ein, und drücken Sie *Return*, um dem Menü *Datei* die Option *Speichern* hinzuzufügen.

6 Geben Sie für den nächsten Eintrag im Menü *Datei* einen Gedankenstrich ein, und bestätigen Sie mit *Return*. Dadurch wird ein Trennstrich in das Menü eingefügt.

7 Geben Sie `&Beenden` ein, und bestätigen Sie mit *Return*, um dem Menü *Datei* die Option *Beenden* hinzuzufügen.

8 Klicken Sie auf den zweiten Hauptmenüeintrag (rechts neben *Datei*). Geben Sie `&Bearbeiten` ein, und drücken Sie *Return*. Dadurch wird der Eintrag unter *Bearbeiten* ausgewählt.

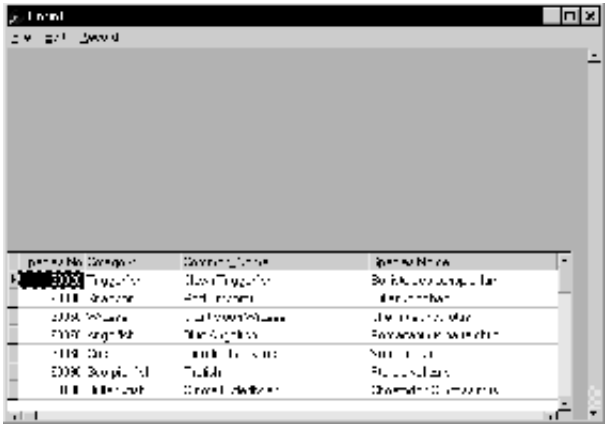
- Setzen Sie im Objektinspektor die Eigenschaft *Action* dieses Eintrags auf *EditCut1*.
- Wählen Sie den Eintrag unter *Ausschneiden* aus, und setzen Sie seine Eigenschaft *Action* auf *EditCopy1*.
- Wählen Sie den Eintrag unter *Kopieren* aus, und setzen Sie seine *Action*-Eigenschaft auf *EditPaste1*.

9 Klicken Sie auf den dritten Hauptmenüeintrag (rechts neben *Bearbeiten*). Geben Sie als Bezeichnung des Elements `D&atensatz` ein, und bestätigen Sie mit *Return*. Dadurch wird der Eintrag unter *Datensatz* ausgewählt.

- Setzen Sie im Objektinspektor die Eigenschaft *Action* dieses Eintrags auf *First1*.
- Wählen Sie den Eintrag unter *Erster* aus, und setzen Sie seine Eigenschaft *Action* auf *Prior1*.
- Wählen Sie den Eintrag unter *Vorheriger* aus, und setzen Sie seine Eigenschaft *Action* auf *Next1*.
- Wählen Sie den Eintrag unter *Nächster* aus, und setzen Sie seine *Action*-Eigenschaft auf *Last1*.

10 Schließen Sie den Menü-Designer.

Sie können jetzt das Programm mit *F9* starten, um sich seine Benutzeroberfläche anzusehen.



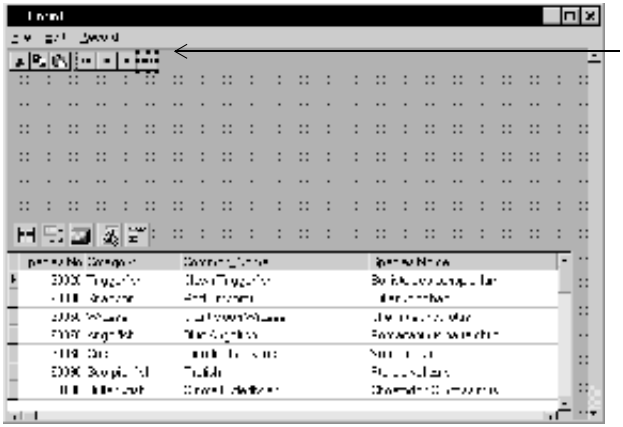
Schließen Sie die Anwendung, wenn Sie weitermachen wollen.

Eine Symbolleiste hinzufügen

1. Doppelklicken Sie in der Registerkarte *Win32* der Komponentenpalette auf die *ToolBar*-Komponente. Dadurch wird im Formular unterhalb des Hauptmenüs automatisch eine Symbolleiste eingefügt.
   - Geben Sie für die Eigenschaft *Indent* den Wert *4* ein.
   - Setzen Sie die Eigenschaft *Images* auf *ImageList1*.
   - Setzen Sie die Eigenschaft *ShowHint* auf *True*.
2. Fügen Sie nun der Symbolleiste Schaltflächen hinzu:
   - Klicken Sie mit der rechten Maustaste in die Symbolleiste, und wählen Sie *Neuer Schalter*. Wiederholen Sie diese Aktion dreimal.
   - Klicken Sie mit der rechten Maustaste, und wählen Sie *Neuer Trenner*.
   - Fügen Sie mit *Neuer Schalter* vier weitere Schaltflächen hinzu.
3. Weisen Sie der ersten Schaltergruppe Aktionen zu:
   - Wählen Sie die erste Schaltfläche aus, und setzen Sie ihre Eigenschaft *Action* auf *Ausschneiden1*.
   - Wählen Sie die zweite Schaltfläche aus, und setzen Sie ihre Eigenschaft *Action* auf *Kopieren1*.
   - Wählen Sie die dritte Schaltfläche aus, und setzen Sie ihre Eigenschaft *Action* auf *Einfgen1*.
4. Weisen Sie der zweiten Schaltergruppe Aktionen zu:

- Wählen Sie die erste Schaltfläche aus, und setzen Sie ihre Eigenschaft *Action* auf *Erster1*.
- Wählen Sie die zweite Schaltfläche aus, und setzen Sie ihre Eigenschaft *Action* auf *Vorheriger1*.
- Wählen Sie die dritte Schaltfläche aus, und setzen Sie ihre Eigenschaft *Action* auf *Nchster1*.
- Wählen Sie die letzte Schaltfläche aus, und setzen Sie ihre Eigenschaft *Action* auf *Letzter1*.

Die Symbolleiste sieht jetzt folgendermaßen aus:



Die Symbolleiste ist eingerichtet und verwendet die vordefinierten Standardaktionen.

5 Drücken Sie *F9*, um das Programm zu compilieren und auszuführen.

Testen Sie nun die Symbolleiste. Die Schaltflächen *Erster, Nächster, Vorgehender* und *Letzter* funktionieren bereits, ebenso die Schaltflächen *Ausschneiden, Kopieren* und *Einfügen*. Um das überprüfen zu können, müssen Sie zuerst Text in einer Gitterzelle markieren.

Wenn Sie bereit sind fortzufahren, schließen Sie die Anwendung.

Eine Grafik anzeigen

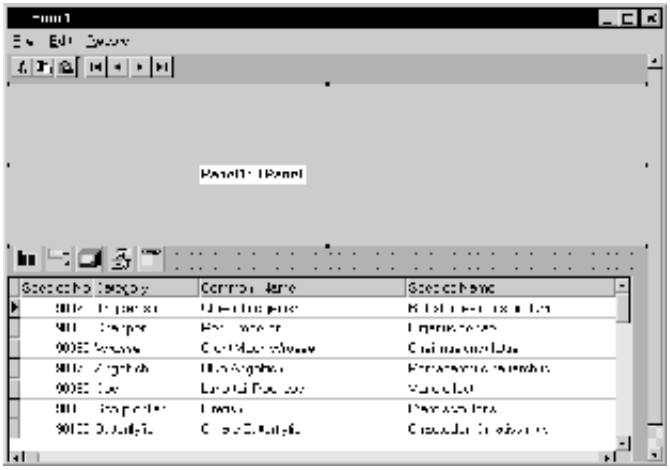
Jedem Datensatz in BIOLIFE.DB ist ein Bild zugeordnet. In diesem Abschnitt werden Sie dafür sorgen, daß diese Bilder angezeigt werden können.

1 Aktivieren Sie die Registerkarte *Standard* der Komponentenpalette, und plazieren Sie eine *Panel*-Komponente unterhalb der Symbolleiste. Die Komponente erhält automatisch den Namen *Panel1*.

2 Löschen Sie im Objektinspektor den Inhalt der Eigenschaft *Caption* (*Panel1*), damit der Titel zur Laufzeit nicht angezeigt wird.

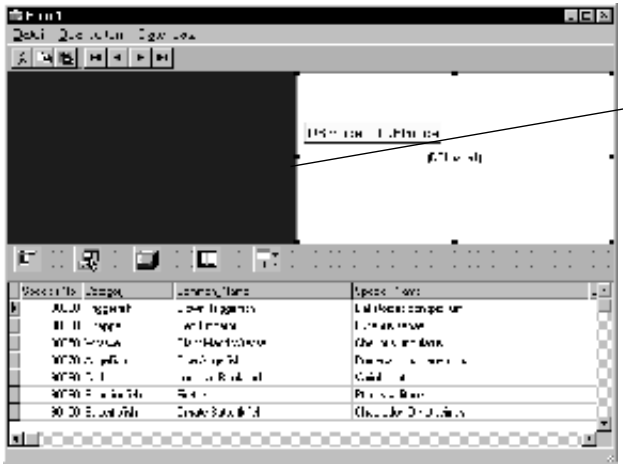
3 Richten Sie die Komponente *Panel1* am oberen Rand des Formulars aus, indem Sie ihre Eigenschaft *Align* auf *alTop* setzen. Ziehen Sie anschließend den unteren Rand

der Tafel so weit nach unten, bis die Tafel den gesamten oberen Bereich des Formulars einnimmt.



4 Setzen Sie die Eigenschaft *Color* der Tafel auf *clBlue.*

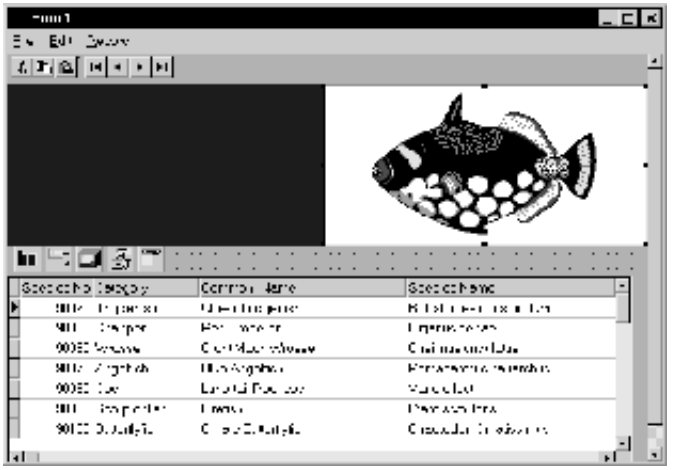
5 Plazieren Sie in *Panel1* eine *DBImage*-Komponente aus der Registerkarte *Datensteuerung*. Setzen Sie die Eigenschaft *Align* dieser Komponente auf *alRight*, und ziehen Sie ihren linken Rand, bis das Formular aussieht wie in der folgenden Abbildung:



Sie können die Breite von *DBImage* durch Ziehen oder im Objektinspektor mit Hilfe der Eigenschaft *Width* ändern.

6 Setzen Sie die Eigenschaft *DataSource* von *DBImage* auf *DataSource1*. Dann setzen Sie die Eigenschaft *DataField* von *DBImage* auf *Graphic*. (Die Dropdown-Liste, die im Objektinspektor zu *DataField* gehört, enthält alle Felder von BIOLIFE.DB. *Graphic* ist einer der Feldnamen.)

Sobald Sie *DataField* auf *Graphic* gesetzt haben, wird das Bild des im ersten Datensatz gespeicherten Fisches angezeigt. Daran erkennen Sie, daß die Verbindung mit der Datenbank korrekt eingerichtet ist.



7 Drücken Sie *F9*, um das Programm zu compilieren und auszuführen.

Wenn Sie bereit sind fortzufahren, schließen Sie die Anwendung.

Text- und Memoobjekte hinzufügen

In diesem Abschnitt werden Sie zwei Komponenten hinzufügen, mit deren Hilfe sich Textfelder aus der Datenbank anzeigen lassen.

1 Wählen Sie *Panel1* aus.

2 Aktivieren Sie die Registerkarte *Datensteuerung*, und fügen Sie in *Panel1* eine *DBMemo*-Komponente ein. Plazieren Sie das Memofeld so, daß es den oberen linken Bereich der Tafel einnimmt (unterhalb der Menü- und Symbolleiste).

3 Verändern Sie die Größe der *DBMemo*-Komponente durch Ziehen ihrer rechten unteren Ecke. Die rechte Kante der Komponente soll die linke Kante der *DBImage*-Komponente berühren, während ihre Unterkante etwa in die Position gebracht werden soll, die der nachfolgenden Abbildung entspricht.

4 Legen Sie im Objektinspektor für *DBMemo* folgende Eigenschaften fest:

- Setzen Sie *DataSource* auf *DataSource1*.
- Setzen Sie *DataField* auf *Notes* (Informationen über den jeweils angezeigten Fisch).
- Setzen Sie *ScrollBars* auf *ssVertical*.

5 Fügen Sie unterhalb des *DBMemo*-Objekts eine *DBText*-Komponente hinzu. Vergrößern Sie die Komponente, bis sie diesen Bereich vollständig ausfüllt. Legen Sie anschließend folgende Eigenschaften der neuen Komponente fest:

- Setzen Sie *DataSource* auf *DataSource1*.
- Setzen Sie *DataField* auf *Common_Name*.
- Setzen Sie *Alignment* auf *taCenter*.

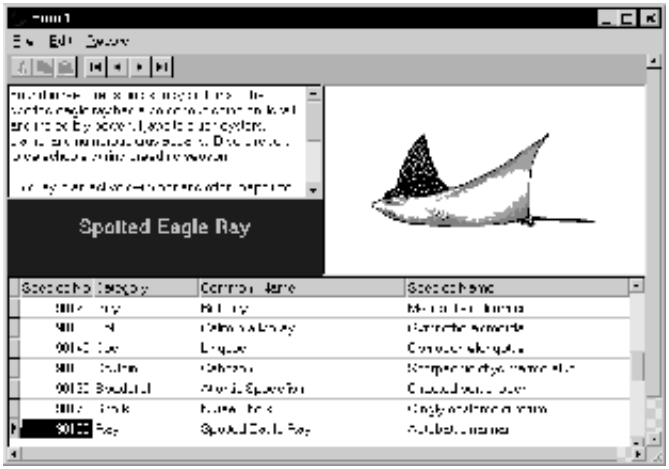
6 Ändern Sie die Eigenschaft *Font* des *DBText*-Objekts mit Hilfe des Schrifteditors.

Der Schrifteditor ist ein Eigenschaftseditor, auf den Sie über den Objektinspektor zugreifen können. Dazu wählen Sie im Objektinspektor die Eigenschaft *Font*. Rechts neben dem Eigenschaftswert wird eine Ellipsen-Schaltfläche angezeigt. Sobald Sie diese Schaltfläche anklicken, erscheint der Schrifteditor.

Nehmen Sie im Schrifteditor (Dialogfeld *Schriftart*) folgende Einstellungen vor, und klicken Sie anschließend auf *OK*:

- Setzen Sie *Schriftschnitt* auf *Fett*.
- Setzen Sie *Farbe* auf *Grau*.
- Setzen Sie *Grad* auf *12*.

7 Compilieren und starten Sie Ihre Anwendung mit *F9*.



Die in der *DBMemo*-Komponente angezeigten Daten können auch bearbeitet werden. Die in der *DBText*-Komponente angezeigten Daten hingegen lassen sich nicht verändern.

Wenn Sie bereit sind fortzufahren, schließen Sie die Anwendung.

Eine Ereignisbehandlungsroutine erstellen

Bisher haben Sie noch keine einzige Zeile Quelltext geschrieben. So sollte es auch sein, wenn Sie von Delphi wirklich profitieren wollen. Sie legen im Objektinspektor die Entwurfszeitwerte der Objekteigenschaften fest und überlassen es Delphi, den generierten Quelltext zu pflegen. In diesem Abschnitt aber werden Sie Prozeduren schreiben, die als Ereignisbehandlungsroutinen bezeichnet werden. Diese Routinen ermöglichen während der Ausführung eines Programms Reaktionen auf Benutzereingaben. Sie werden diese Ereignisbehandlungsroutinen mit Menübefehlen verknüpfen, so daß bei der Auswahl eines dieser Befehle Ihre Anwendung den Code der damit verknüpften Routine ausführt.

1. Öffnen Sie die Registerkarte *Dialoge* der Komponentenpalette, und fügen Sie dem Formular eine *SaveDialog*-Komponente hinzu. Da es sich um eine nichtvisuelle Komponente handelt, ist ihre Position auf dem Formular ohne Bedeutung. Delphi weist der Komponente automatisch den Standardnamen *SaveDialog1* zu. (Wird die Methode *Execute* dieser Komponente aufgerufen, führt dies zum Aufruf des Standard-Windows-Dialogfelds zum Speichern von Dateien.)
2. Wählen Sie aus dem Menü Ihres Formulars den Befehl *Datei / Speichern*. Delphi erstellt nun eine Standard-Ereignisbehandlungsroutine für den Fall, daß der Benutzer zur Laufzeit den *Speichern*-Befehl aus dem Menü *Datei* wählt. Der Quelltexteditor wird geöffnet, und der Cursor befindet sich innerhalb des Quelltextes.

Diese Ereignisbehandlungsroutine ist mit dem *OnClick*-Ereignis des ersten Menübefehls des Hauptmenüs verknüpft. Dieser Menübefehl ist eine Instanz der Klasse *TMenuItem*, und *OnClick* ist das zugehörige *Standardereignis*. Folglich ist die Methode *Save1Click* eine *Standard-Ereignisbehandlungsroutine*.

3 Vervollständigen Sie nun die Ereignisbehandlungsroutine, indem Sie den nachfolgend wiedergegebenen Quelltext in den **var**-Abschnitt und zwischen die äußeren **begin**- und **end**-Anweisungen einfügen.

```
procedure TForm1.Save1Click(Sender: TObject);
  var
    i: integer;
  begin
    SaveDialog1.Title := Format('Save info for %s', [DBText1.Field.AsString]);
    if SaveDialog1.Execute then
    begin
      with TStringList.Create do
      try
        Add(Format('Facts on the %s', [DBText1.Field.AsString]));
        Add(#13#10);
        for i := 1 to DBGrid1.FieldCount-3 do
        Add(Format('%s : %s',
          [DBGrid1.Fields[i].FieldName,
          DBGrid1.Fields[i].AsString]));
        Add(Format(#13#10+'%s'+#13#10,[DBMemo1.Text]));
        SaveToFile(SaveDialog1.FileName);
      finally
        Free;
      end;
    end;
  end;
```

Diese Ereignisbehandlungsroutine ruft die Methode *Execute* der Komponente *SaveDialog* auf. Wenn sich das Dialogfeld öffnet und der Benutzer einen Dateinamen angibt, werden die Felder der aktuellen Datenbank in einer Datei gespeichert.

4 Um Code für den Befehl *Beenden* hinzuzufügen, wählen Sie *Datei / Beenden*. Delphi generiert nun eine weitere Standard-Ereignisbehandlungsroutine und zeigt sie im Editor an.

```
procedure TForm1.Exit1Click(Sender: TObject);
begin

end;
```

Geben Sie an der Cursorposition (zwischen **begin** und **end**) folgendes ein:

```
Close;
```

5 Speichern Sie Ihre Arbeit mit *Datei / Alles Speichern*. Drücken Sie dann *F9*, um die Anwendung zu starten.

Das Programm kann mit dem nunmehr funktionsfähigen Menübefehl *Datei / Beenden* beendet werden.

Die meisten Komponenten der Komponentenpalette verfügen über Ereignisse, und die meisten Komponenten verfügen auch über ein Standardereignis. Oft handelt es sich dabei um das Ereignis *OnClick*, das immer dann ausgelöst wird, wenn die Komponente angeklickt wird. Wenn Sie beispielsweise eine *Button*-Komponente (*TButton*) in ein Formular einfügen, werden Sie dafür höchstwahrscheinlich auch

eine *OnClick*-Ereignisbehandlungsroutine schreiben. Wenn Sie gewisse Objekte in einem Formular doppelt anklicken, erstellt Delphi für das zugehörige Standardereignis eine Standard-Ereignisbehandlungsroutine.

Auch über den Objektinspektor können Sie auf die Ereignisse einer Komponente zugreifen. Dazu markieren Sie ein auf einem Formular befindliches Objekt und öffnen die Registerkarte *Ereignisse* des Objektinspektors. Es erscheint nun eine Liste mit den Ereignissen des Objekts. Um für ein beliebiges Ereignis eine Standard-Ereignisbehandlungsroutine zu erstellen, klicken Sie die Leerfläche rechts neben dem betreffenden Ereignis doppelt an.

Weitere Informationen über Ereignisse und Ereignisbehandlungsroutinen finden Sie im Kapitel »Die Benutzeroberfläche erstellen« im *Entwicklerhandbuch* oder in der Online-Hilfe.

Kapitel

4

Die Umgebung anpassen

In diesem Kapitel erfahren Sie, wie Sie die Delphi-Umgebung Ihren Bedürfnissen anpassen können.

Den Arbeitsbereich organisieren

Die IDE stellt zahlreiche Tools zur Verfügung, die die Anwendungsentwicklung vereinfachen, so etwa den Formular-Designer, den Objekinspektor, den Quelltexteditor, die Projektverwaltung, den Projekt-Browser und die verschiedenen Debugger-Fenster. Bei einer solchen Fülle an Tools ist es ratsam, den Arbeitsbereich möglichst gut zu organisieren.

Tool-Fenster andocken

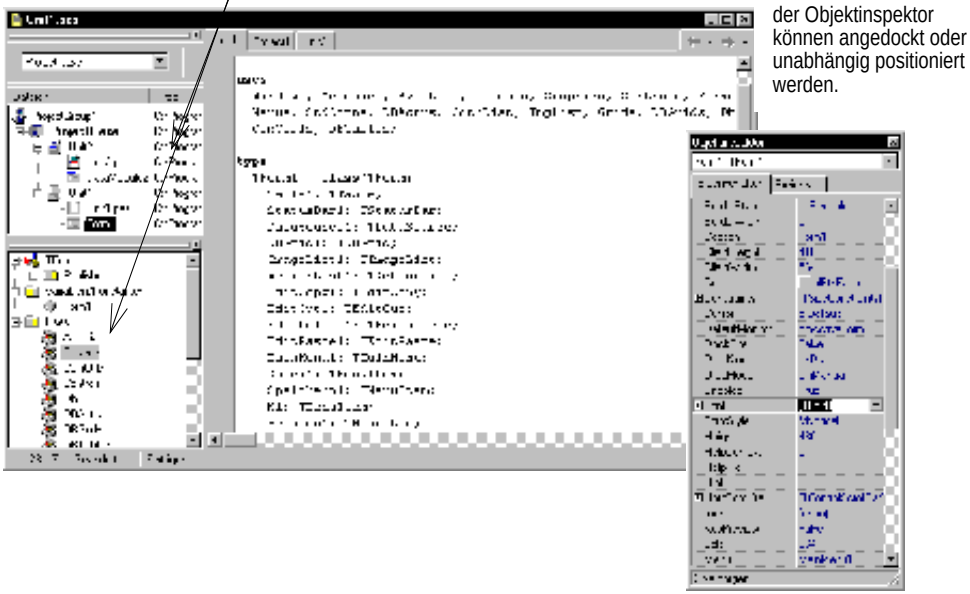
In Delphi können die verschiedenen Tool-Fenster beliebig auf dem Desktop angeordnet und unabhängig voneinander geöffnet und geschlossen werden. Viele dieser Fenster können auch an andere Fenster *angedockt* werden. Darunter ist zu verstehen, daß mehrere Fenster fest miteinander verbunden werden können, so daß sie eine Einheit bilden, oder daß verschiedene Fenster in einem mehrseitigen Dialogfeld zusammengefaßt und über ein Register geöffnet 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 beim Schreiben von Quelltext oder beim Debuggen einer Anwendung direkt an den Quelltexteditor andocken. Dazu ein Beispiel: Wenn Sie Delphi zum ersten Mal aufrufen (ohne also die Standardkonfiguration geändert zu haben), ist auf der linken Seite des Quelltextedi-

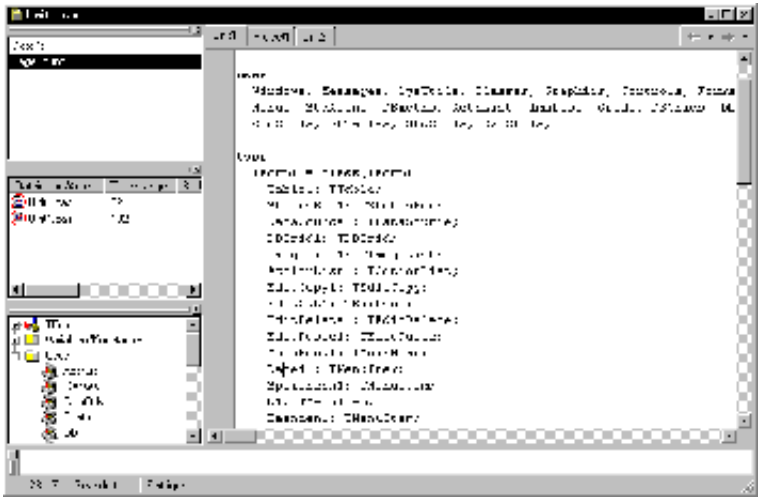
tors der Code-Explorer angedockt. Bei Bedarf können Sie an diese beiden Fenster auch die Projektverwaltung andocken.

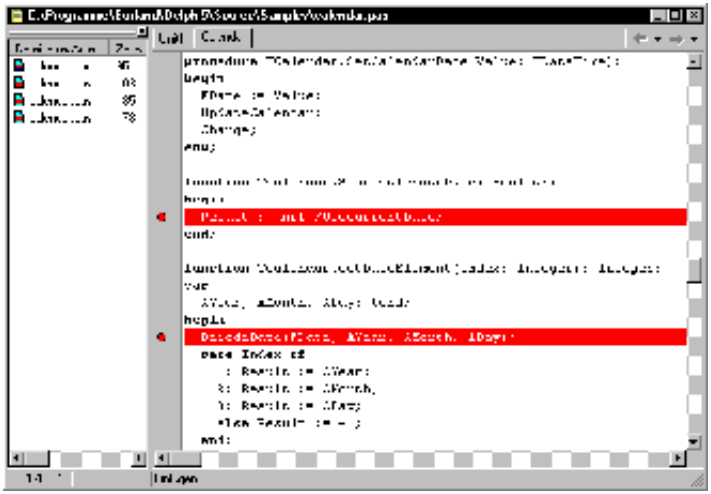
Hier sind die Projektverwaltung und der Klassen-Explorer an den Quelltexteditor angedockt.

Auch andere Tools wie der Objektinspektor können angedockt oder unabhängig positioniert werden.



Beim Debuggen einer Anwendung können Sie das Fenster mit den überwachten Ausdrücken und das Haltepunkt-Fenster an den Quelltexteditor andocken.





Hier wurde nur das Haltepunkt-Fenster an den Quelltexteditor angedockt.

Sie können auch mehrere Tools in Form eines mehrseitigen Dialogfeldes mit Registerkarten zusammenfassen.

Hier wurden mehrere Fenster des Debuggers in ein mehrseitiges Dialogfeld integriert.

Um ein Fenster anzudocken, ziehen Sie es über ein anderes Fenster, bis sein rechtekkiger Umriß schmal wird. Lassen Sie dann die Maustaste los. Um eine Andockung aufzuheben, klicken Sie die Titelleiste des betreffenden Fensters an und ziehen es in eine beliebige Richtung.

Weitere Informationen

Siehe im Hilfeindex unter »Andocken«.

Menüs und Symbolleisten organisieren

Das Hauptfenster nimmt den oberen Bereich des Bildschirms ein und enthält das Menü, die Symbolleisten und die Komponentenpalette. Sie können seinen Inhalt beliebig anordnen.

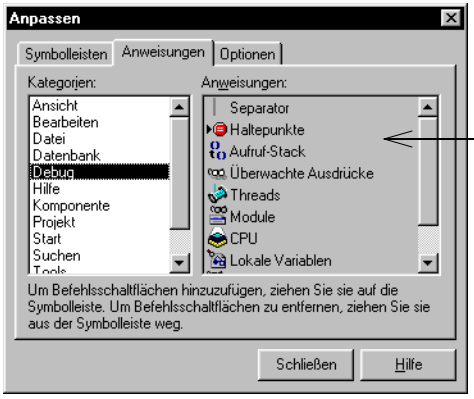
Das Hauptfenster in der Standardanordnung.

Sie können die Menüs und Symbolleisten innerhalb des Hauptfensters beliebig verschieben. Ziehen Sie einfach die Griffleiste (der schmale, geriffelte Bereich an der linken Seite des Menüs und der Symbolleisten) an die gewünschte Position.

Das umgestaltete Hauptfenster

Sie können diese Elemente sogar vom Hauptfenster trennen, beliebig anordnen oder ganz vom Desktop entfernen.

Sie können auch die Symbolleisten individuell anpassen, indem Sie Tools hinzufügen oder entfernen.



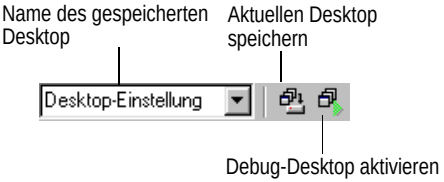
Klicken Sie in einer Werkzeugleiste mit der rechten Maustaste, und wählen Sie *Anpassen.*

Ziehen Sie die gewünschten Befehle aus dem Feld *Anweisungen* auf die Werkzeugleiste.

Desktop-Einstellungen anpassen

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 zur Verfügung. Die Auswahl des jeweils gewünschten Desktops erfolgt dann mit Hilfe von zwei Symbolschaltern.

Richten Sie den Desktop in der gewünschten Form ein. Dazu gehört sein Aussehen, die Größe der einzelnen Elemente und die Andockpositionen einzelner Fenster. Und dann plazieren Sie alles dort, wo es angezeigt werden soll. Zuletzt klicken Sie in der Symbolleiste *Desktop* den Symbolschalter *Aktuellen Desktop speichern* an oder wählen *Ansicht / Desktop / Desktop speichern*.



Weitere Informationen

Suchen Sie im Hilfeindex nach »Desktop-Layout«.

Standard-Projektoptionen festlegen

Mit Hilfe des Dialogfeldes *Projektoptionen*, auf das Sie über *Projekt / Optionen* zugreifen, werden die Einstellungen festgelegt, die jeweils für eine bestimmte Anwendung gelten. (Siehe dazu »Projekt- und Umgebungsoptionen einstellen« auf Seite 2-12.) Wenn Sie jedoch in der linken unteren Ecke des Dialogfeldes das Kontrollkästchen *Vorgabe* aktivieren, gelten die von Ihnen getroffenen Einstellungen standardmäßig auch für alle neuen Projekte.

Ist das Kontrollkästchen *Vorgabe* aktiviert, werden die aktuellen Einstellungen in die Optionendatei DEFPROJ.DOF geschrieben. Um die Standardeinstellungen von Delphi wiederherzustellen, müssen Sie die Datei DEFPROJ.DOF löschen oder umbenennen. Sie befindet sich im Verzeichnis \BIN von Delphi.

Standardprojekte und -formulare festlegen

Wenn Sie *Datei / Neue Anwendung* wählen, wird in der IDE ein neues Projekt geöffnet. Wenn Sie kein Standardprojekt spezifiziert haben, wird statt dessen für die neue Anwendung nur ein leeres Formular erzeugt. Sie haben jedoch die Möglichkeit, jedes Element, das in der Objektablage auf der Registerkarte *Projekte* aufgeführt ist (siehe »Objekte als Vorlagen speichern« auf Seite 2-19) als Ihr *Standardprojekt* zu definieren. Sobald Sie ein solches Standardprojekt angegeben haben, wird es bei der Auswahl von *Datei / Neue Anwendung* als Vorlage benutzt. Wenn Sie als Standardprojekt einen

Experten angegeben haben, wird bei der Auswahl von *Datei / Neue Anwendung* der betreffende Experte gestartet, der das neue Projekt dann auf der Grundlage Ihrer nachfolgenden Angaben erstellt.

Auf dieselbe Weise können Sie auch einen Standard für ein Hauptformular und einen Standard für ein neues Formular festlegen. Das *Standard-Hauptformular* ist das Formular, das zu Beginn einer neuen Anwendung erstellt wird. Die Festlegung eines Standards für *neue Formulare* hingegen wird immer dann wirksam, wenn Sie *Datei / Neues Formular* wählen, um in ein geöffnetes Projekt ein neues Formular einzufügen. Wenn Sie kein Standardformular definiert haben, benutzt Delphi ein leeres Formular.

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

Weitere Informationen

Suchen Sie im Hilfeindex nach »Projekte, Vorgaben« und »Formulare, Vorgaben«.

Einstellungen für Tools festlegen

Im Dialogfeld *Umgebungsoptionen*, auf das Sie mit *Tools / Umgebungsoptionen* zugreifen, werden zahlreiche Einstellungen festgelegt, die das Aussehen und das Verhalten der IDE steuern. Alle Änderungen, die Sie in diesem Dialogfeld vornehmen, haben globale Auswirkungen, beeinflussen also nicht nur das aktuelle Projekt, sondern auch Projekte, die zu einem späteren Zeitpunkt geöffnet und compiliert werden.

Weitere Informationen

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

Den Quelltexteditor anpassen

Ein Tool, das in der Regel individuell angepaßt wird, ist der Quelltexteditor. Die entsprechenden Optionen sind auf verschiedenen Registerkarten des Dialogfelds *Tools / 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 Quelltexteditor mittels der Registerkarte *Programmierhilfe* des Dialogfelds *Editor-Optionen* zugreifen können, lassen sich individuell konfigurieren. Diese Tools sind im Abschnitt »Hilfe beim Programmieren« auf Seite 2-15 beschrieben.

Weitere Informationen

Klicken Sie auf folgenden Registerkarten des Dialogfeldes *Editor-Optionen* die Schaltfläche *Hilfe* an: *Editor*, Anzeige, *Tastaturbelegung*, *Farben* und *Programmierhilfe*.

Den Formular-Designer anpassen

Auf der Registerkarte *Präferenzen* des Dialogfeldes *Umgebungsoptionen* werden Einstellungen vorgenommen, die den Formular-Designer betreffen. Dort können Sie beispielsweise die Ausrichtfunktion des Rasters aktivieren oder deaktivieren.

Weitere Informationen

Klicken Sie auf der Registerkarte *Präferenzen* des Dialogfeldes *Umgebungsoptionen* die Schaltfläche *Hilfe* an.

Optionen für den Code-Explorer festlegen

Beim Start von Delphi wird automatisch der Code-Explorer geöffnet (siehe »Der Code-Explorer« auf Seite 2-8). Sie können dies verhindern—und zugleich auch andere Optionen für den Code-Explorer festlegen—, indem Sie auf der Registerkarte

Explorer des Dialogfelds *Umgebungsoptionen* die entsprechenden Einstellungen vornehmen.

Hier können Sie auch das anfängliche Aussehen des Code-Browsers sowie dessen Anzeigebereich festlegen. Die betreffenden Einstellungen wirken sich auf die Registerkarte *Explorer* des Dialogfelds *Umgebungsoptionen* aus. Eine Darstellung des Code-Browsers finden Sie unter »Elemente und Struktur eines Projekts anzeigen« auf Seite 2-10.

Weitere Informationen

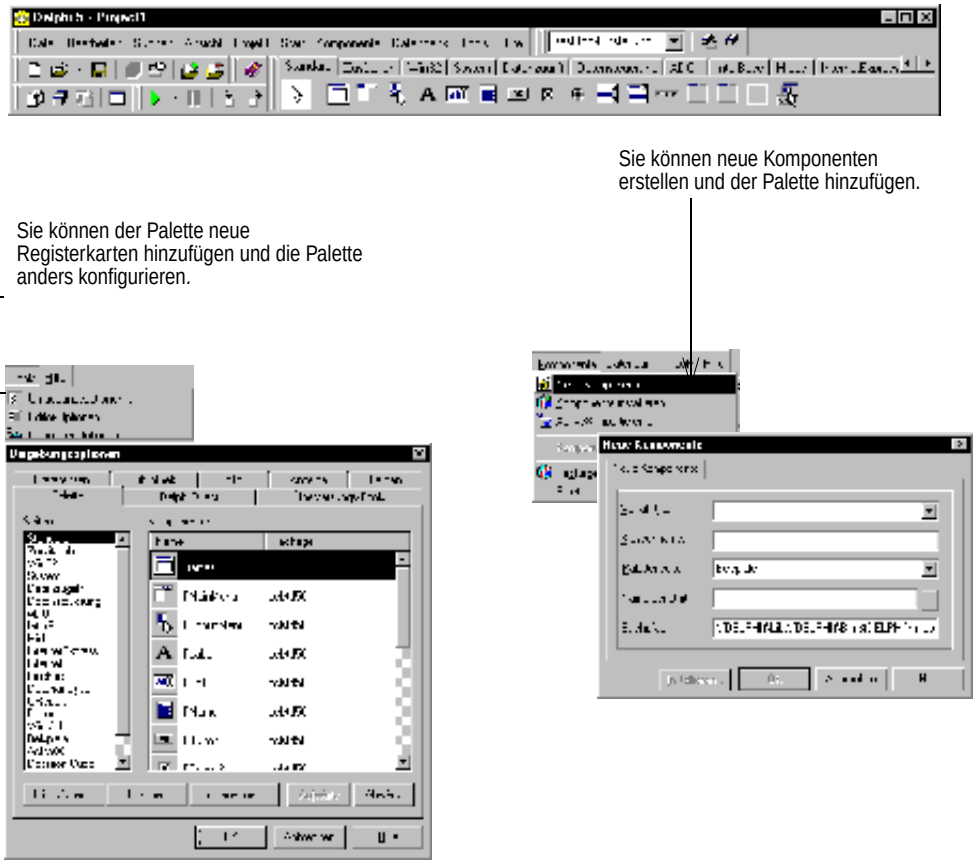
Klicken Sie auf der Registerkarte *Explorer* des Dialogfelds *Umgebungsoptionen* die Schaltfläche *Hilfe* an.

Die Komponentenpalette anpassen

In der Standardkonfiguration von Delphi sind die in der Komponentenpalette enthaltenen VCL-Objekte, geordnet nach ihren jeweiligen Funktionen, auf verschiedene Registerkarten verteilt. Sie haben jedoch die Möglichkeit, die vorgegebene Systematik Ihren individuellen Anforderungen anzupassen.

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

- Sie können Komponentenschablonen erstellen und zur Palette hinzufügen.



Die Komponentenpalette anders konfigurieren

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

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

Weitere Informationen

Klicken Sie im Dialogfeld *Paletteneigenschaften* die Schaltfläche *Hilfe* an.

Komponenten hinzufügen

Die in der VCL enthaltenen Komponenten können Sie durch weitere Komponenten ergänzen, die Sie entweder selbst geschrieben oder von einem Drittanbieter bekommen haben. Damit Sie zur Entwurfszeit auf solche Komponenten zugreifen können, müssen Sie sie vorher in der IDE installieren.

Weitere Informationen

Beachten Sie bei der Installation von Fremdkomponenten die Angaben des Herstellers / Händlers. Die Erstellung eigener Komponenten wird im *Entwicklerhandbuch* bzw. in der Online-Hilfe unter »Benutzerdefinierte Komponenten erstellen« beschrieben.

ActiveX-Steuerelemente hinzufügen

Sie können in die Komponentenpalette auch ActiveX-Steuerelemente einfügen und sie in Ihren Delphi-Projekten verwenden. Dazu wählen Sie *Komponente / ActiveX importieren*. Dadurch wird das Dialogfeld *ActiveX-Element importieren* geöffnet, in dem Sie neue ActiveX-Elemente registrieren lassen können oder ein bereits registriertes Element zur Installation auswählen können. Bei der Installation eines ActiveX-Steuerelements erstellt und compiliert Delphi dafür als »Verpackung« eine Unit-Datei.

Weitere Informationen

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

Komponentenschablonen erstellen

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

Nachdem eine solche Schablone 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.

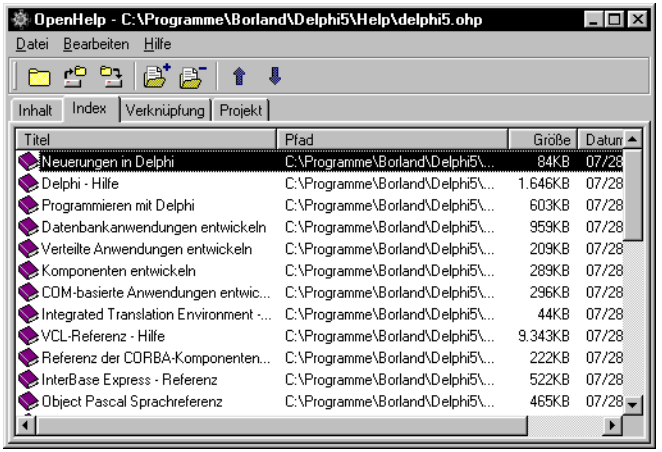
Weitere Informationen

Suchen Sie im Hilfeindex nach »Vorlage« bzw. nach »Schablone«, oder wählen Sie *Komponente / Komponentenvorlage erzeugen*, und drücken Sie dann *F1*.

Das Hilfesystem anpassen

Das Hilfesystem von Delphi enthält mehr als ein Dutzend Windows-Hilfedateien (HLP). Es umfaßt eine komplette Dokumentation der IDE, der Visual Component Library und anderer Produkte und Tools, die im Lieferumfang von Delphi enthalten sind. Ein Dienstprogramm namens *OpenHelp* gibt Ihnen die Möglichkeit, das Hilfesystem individuell anzupassen. Damit können Sie festlegen, auf welche Hilfedateien über das Inhaltsverzeichnis, den Index und die kontextsensitive Hilfe der IDE zugegriffen werden kann.

OpenHelp wird mit *Hilfe / Anpassen* gestartet.



Das Standard-Hilfesystem von Delphi wird durch die Datei BCB4.OHP im Verzeichnis HELP konfiguriert. Diese Datei kann durch Hinzufügen oder Entfernen von Hilfedateien angepaßt werden.

Diese Liste bestimmt, welche Hilfedateien im Inhaltsverzeichnis des Hilfesystems angezeigt werden.

Mit *OpenHelp* können Sie beliebige WinHelp-Dateien in das Hilfesystem von Delphi aufnehmen, einschließlich Hilfedateien externer Produkte. Außerdem können Sie damit Referenzen auf obsolete Hilfedateien aus der Systemregistrierung entfernen.

Einen Überblick über die Hilfedateien von Delphi erhalten Sie im Abschnitt »Online-Hilfe« auf Seite 1-2.

Weitere Informationen

Wählen Sie zunächst *Hilfe / Anpassen*. Im Hauptfenster von *OpenHelp* wählen Sie dann *Hilfe / Inhalt*.

Kapitel

5

Mit Delphi programmieren

Dieses Kapitel gibt einen Überblick über die Software-Entwicklung mit Delphi und beschreibt einige Features, die in der vorliegenden *Einführung* noch nicht behandelt wurden.

Die Tools der Entwicklungsumgebung

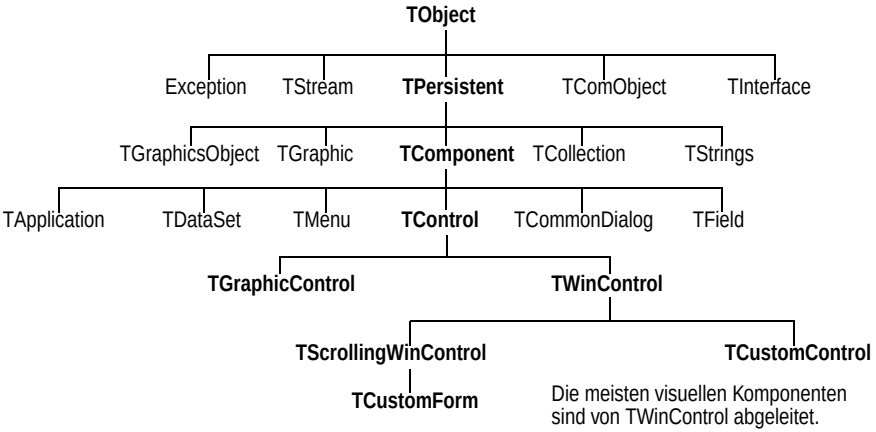
Zur integrierten Entwicklungsumgebung (IDE) von Delphi gehören der Formular-Designer, der Objektinspektor, die Komponentenpalette, die Projektverwaltung, der Code-Explorer, der Quelltexteditor, der Datenmodul-Designer; Tools für die Software-Loklaisierung, der Debugger sowie zahlreiche andere Tools. Welche Features und Komponenten Ihnen jeweils zur Verfügung stehen, hängt davon ab, mit welcher Version von Delphi Sie arbeiten.

Sämtliche Versionen von Delphi unterstützen die allgemeine 32-Bit-Programmierung unter Windows, Multithreading, COM sowie Automatisierungs-Controller und Multiprozeß-Debugging. Einige Versionen bieten zusätzlich Unterstützung für Serveranwendungen wie COM-Server und Web-Anwendungen, für die Entwicklung von Datenbankanwendungen mit Report- und Chart-Ausgabe für eine Reihe von DBMS-Backends, für SQL-Datenbankserver (wie etwa Oracle 8 and InterBase), schließlich für Microsoft Transaction Server (MTS), für mehrschichtige Datenbankanwendungen, für CORBA sowie für Decision-Support.Systeme. Aktuelle Produktinformationen erhalten Sie von Ihrem Inprise-Händler oder unter www.borland.com.

Die VCL verwenden

Delphi wird mit Komponenten ausgeliefert, die Teil einer Klassenhierarchie, der sogenannten Visual Component Library (VCL), sind. Die VCL enthält sowohl Objekte, die zur Laufzeit sichtbar sind, also Editierfelder, Schaltflächen und andere Elemente der Benutzerschnittstelle, als auch unsichtbare Steuerelemente wie Datenmengen

und Timer. Die folgende Abbildung zeigt die Beziehungen zwischen einigen dieser Klassen.



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

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

Die meisten Versionen von Delphi werden mit dem kompletten VCL-Quellcode ausgeliefert. Zusätzlich zur Online-Dokumentation enthält dieser Quellcode eine Reihe von wertvollen Beispielen für die Object-Pascal-Programmierung.

Weitere Informationen

Siehe in der Online-Hilfe unter »VCL-Referenz« und »Benutzerdefinierte Komponenten erzeugen«.

Exceptions behandeln

Fehlerbedingungen werden in Delphi durch Exceptions abgefangen. Das sind besondere Objekte, die als Reaktion auf unvorhergesehene Eingaben oder fehlerhafte Programmabläufe generiert werden. Exceptions können sowohl zur Entwurfs- als auch zur Laufzeit ausgelöst werden. In der VCL sind zahlreiche Exception-Klassen enthalten, die mit speziellen Fehlerbedingungen verknüpft sind. Um Laufzeitfehlern erfolgreich begegnen zu können, werden Sie für Ihre Anwendungen Routinen zur *Exception-Behandlung* schreiben. Exceptions können auch ein wertvolles Debugging-Tool sein, da ja die Klasse einer Exception oft darüber Auskunft gibt, durch welche Art von Fehler sie ausgelöst wurde.

Weitere Informationen

Siehe in der Online-VCL-Referenz unter »Exception«. Beachten Sie auch die Einträge zu den spezialisierten abgeleiteten Klassen. Suchen Sie außerdem im Hilfeindex nach »Exception-Behandlung«.

Datenbank-Connectivity und Datenbank-Utilities

Um die Entwicklung von Datenbankanwendungen zu vereinfachen, stellen Delphi und die VCL eine Reihe von Connectivity-Tools zur Verfügung. Die Borland Database Engine (BDE) ist eine Sammlung von Treibern für zahlreiche populäre Datenbankformate. Dazu gehören dBASE, Paradox, FoxPro, 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.

Delphi enthält Komponenten, die einen Datenzugriff mittels InterBase Express (IBX) ermöglichen. IBX-Anwendungen gewähren Zugriff auf fortgeschrittene InterBase-Features und bieten die derzeit leistungsfähigste Komponentenschnittstelle für InterBase 5.5 oder neuere Versionen.

IBX basiert auf der Delphi-Komponentenarchitektur für individuell angepaßten Datenzugriff und bildet eine funktionelle Einheit mit dem Datenmodul-Designer. IBX ist kompatibel mit der Bibliothek der datensensitiven Steuerelemente von Delphi und erfordert nicht die Verwendung der Borland Database Engine (BDE).

Im Formular-Designer können Sie zur Entwurfszeit Datenbanktabellen einrichten. Hierbei erstellen Sie zunächst mit Hilfe des Objektinspektors Felddefinitionen, klicken dann die Tabellenkomponente mit der rechten Maustaste an und wählen *Tabelle erzeugen*.

Einige Delphi-Versionen enthalten Komponenten, die eine Verknüpfung mit Datenbanken über ActiveX-Datenobjekte (ADO) ermöglichen. ADO ist eine von Microsoft entwickelte High-level-Schnittstelle zu sämtlichen Datenquellen, einschließlich relationalen und nicht-relationalen Datenbanken, E-mail- und Dateisystemen, Text und Grafiken sowie zu individuellen Business-Objekten.

Weitere Informationen

Siehe im *Entwicklerhandbuch* oder in der Online-Hilfe unter »Datenbankanwendungen entwickeln«.

Zusätzlich stehen für Datenbankentwickler folgende Tools zur Verfügung.

BDE-Administrator

Der BDE-Administrator (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

Starten Sie den BDE-Administrator von der Delphi-Programmgruppe aus. Wählen Sie dann *Hilfe / Inhalt*.

SQL-Explorer (Datenbank-Explorer)

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

Weitere Informationen

Öffnen Sie den Datenbank-Explorer, indem Sie im Delphi-Hauptmenü *Datenbank / Explorer* wählen; drücken Sie dann *F1*. Oder suchen Sie im Hauptindex des Hilfesystems nach »Datenbank-Explorer«.

Datenbank-Desktop

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

Weitere Informationen

Starten Sie den Datenbank-Desktop von der Delphi-Programmgruppe aus. Drücken Sie dann *F1*.

Daten-Dictionary

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

Weitere Informationen

Suchen Sie im Hilfeindex nach »Daten-Dictionary«.

Projektarten

Mit Delphi lassen sich Windows-GUI-Anwendungen, Konsolenanwendungen, Service-Anwendungen, Dynamische Link-Bibliotheken (DLLs), Packages (ein besonderer, von Delphi benutzter Typ von DLL) und andere Programme erstellen.

Anwendungen und Server

Delphi verfügt über Features, die die Erstellung verteilter Anwendungen erleichtern. Dazu gehören Client / Server-, mehrschichtige und Web-basierte Systeme. Zusätzlich zu einer Reihe von Internet-Komponenten und zur Unterstützung von Standards wie COM stellen einige Versionen von Delphi auch Tools zur Entwicklung von CORBA-Anwendungen zur Verfügung.

Weitere Informationen

Siehe im *Entwicklerhandbuch* oder in der Online-Hilfe unter »Anwendungen, Komponenten und Bibliotheken erstellen« und »Verteilte Anwendungen erstellen«.

DLLs

Dynamische Link-Bibliotheken (DLLs) 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.

Weitere Informationen

Suchen Sie im Hilfeindex nach »DLLs«.

Benutzerdefinierte Komponenten und Packages

Ein *Package* ist eine spezielle DLL, die von Delphi-Anwendungen, von der IDE oder von beiden benutzt wird. Zwar können Packages auf verschiedenste Art eingesetzt werden, doch besteht ihr wesentlicher Vorteil in der Kapselung von Delphi-Komponenten. Tatsächlich müssen alle in der IDE installierten Komponenten als Packages compiliert werden.

Die mit Delphi ausgelieferten Komponenten sind in der IDE vorinstalliert und bieten ein Funktionalitätsspektrum, das für die Mehrzahl Ihrer Anwendungen ausreichen dürfte. Sie könnten also mit Delphi jahrelang programmieren, ohne je eine neue Komponente installieren zu müssen. Und doch werden Sie zuweilen das Bedürfnis haben, spezielle Probleme zu lösen oder bestimmte Verhaltensweisen zu kapseln, die benutzerdefinierte Komponenten erfordern.

Benutzerdefinierte Komponenten ergänzen die VCL, machen Code wiederver-wendbar und fördern die Konsistenz von Anwendungen. Zahlreiche Delphi-Komponenten lassen sich von Drittanbietern beziehen, und Delphi stellt einen *Experten für neue Komponenten* zur Verfügung, mit dessen Hilfe es Ihnen ein leichtes ist, benutzerdefinierte Komponenten zu erstellen und zu installieren.

Weitere Informationen

Siehe im *Entwicklerhandbuch* oder in der Online-Hilfe unter »Benutzerdefinierte Komponenten erstellen«. Suchen Sie außerdem im Hilfeindex nach »Packages«.

Frames

Ein Frame (*TFrame*) ist, wie auch ein Formular, ein Container für andere Komponenten. In mancher Hinsicht haben Frames eher die Eigenschaft einer benutzerspezifischen Komponente als Formulare. Zur bequemen Wiederverwendung lassen sie sich in der Komponentenpalette ablegen. Sie können mit Formularen, anderen Frames und anderen Container-Objekten verschachtelt werden.

Nachdem ein Frame erstellt und gespeichert wurde, fungiert er in der Folgezeit als eine Unit und erbt die Änderungen, die an den in ihm enthaltenen Komponenten (einschließlich anderer Frames) vorgenommen werden. Wenn ein Frame in einen anderen Frame oder ein Formular eingebettet ist, erbt er in der Folgezeit die Änderungen, die an dem Frame vorgenommen wurden, von dem er abgeleitet wurde.

Weitere Informationen

Suchen Sie im Hilfeindex nach »Frames« und »TFrame«.

COM und ActiveX

Delphi unterstützt den von Microsoft entwickelten COM-Standard (COM = Component Object Model) und enthält Experten für die Erstellung von ActiveX-Steuerelementen. Auf der Registerkarte *ActiveX* der Komponentenpalette finden Sie Beispiele für ActiveX-Steuerelemente. Auf der Registerkarte *Server* der Komponentenpalette stehen zahlreiche COM-Server-Komponenten zur Verfügung. Sie können diese Komponenten auf dieselbe Weise 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 »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 Objekte zur Verfügung. Mit dem Typbibliotheks-Editor von Delphi können Sie Typbibliotheken erstellen und bearbeiten.

Weitere Informationen

Suchen Sie im Hilfeindex nach »Typbibliotheken«.

Anwendungen vertreiben

Bei der Weitergabe einer Anwendung müssen Sie dafür sorgen, daß die Benutzer alle erforderlichen Dateien erhalten, einschließlich EXE-Dateien, DLLs, Packages und BDE-Treiber. Um diesen Prozeß zu vereinfachen, steht in Delphi eine spezielle Version von *InstallShield Express* zur Verfügung, ein vielbenutztes Tool zur Entwicklung von Installations-Utilities.

Weitere Informationen

Suchen Sie im Hilfeindex nach »Anwendungen weitergeben«.

Anwendungen internationalisieren

Delphi bietet eine Reihe von Features, mit denen Sie Ihre Anwendungen an unterschiedliche Sprachräume anpassen können. Über die VCL wird Unterstützung für Input-Method-Editoren (IMEs) und erweiterte Zeichensätze zur Verfügung gestellt, und Tools wie der Ressourcen-Experte machen es Ihnen leicht, die Lokalisierung Ihrer Projekte vorzubereiten. Um von diesen Features wirklich profitieren zu können, sollten Sie sich in einer möglichst frühen Phase des Entwicklungsprozesses darüber Gedanken machen, was für die Internationali-sierung Ihrer Anwendung erforderlich ist.

Die in einigen Versionen von Delphi verfügbare Integrated Translation Environment (ITE) besteht aus Tools für die Software-Lokalisierung und die gleichzeitige Erstellung unterschiedlicher internationaler Versionen. Sie ist integraler Bestandteil der IDE und gibt Ihnen die Möglichkeit, mehrere lokalisierte Versionen als Teil eines einzigen Projekts zu verwalten.

Die ITE enthält drei Tools:

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

Weitere Informationen

Suchen Sie im Hilfeindex nach »Internationale Anwendungen« und nach »ITE«.

Index

A

Access 5-3
ActionList (Komponente) 3-8
ActiveX 5-6
  Registerkarte der Komponentenpalette 5-6
ActiveX-Steuerelemente 4-10
Aktionslisten 3-8
  Listeneditor 3-8
Anwendungen
  erstellen 3-1, 3-2
  Fehler beheben 2-17
  internationalisieren 5-7
  lokalisieren 5-7
  vertreiben 5-6
Ausdrücke
  auswerten 2-16
Automatisierung 5-1

B

BDE (Borland Database Engine) 5-3
  Administrator 3-7, 5-3
  Aliase 3-7, 5-3
Beispielanwendung 3-1
Beispieldatenbank 3-6
Benutzeroberfläche 2-4, 3-2
Bibliothek visueller Komponenten (VCL) 2-5
BIOLIFE.DB 3-6
Botschaften (Windows) 5-2

C

Charts 5-1
Code *Siehe* Quelltext
Code-Browser 4-8
Code-Explorer 4-7
Code-Parameter 2-15
Code-Vervollständigung 2-15
Code-Vorlagen 2-15
Color (Eigenschaft) 3-3
COM 5-1, 5-4, 5-6
CORBA 5-1, 5-4

D

DataSource (Komponente) 3-5, 3-7
Datei speichern (Dialogfeld) 3-16
Dateien speichern 3-16
Datenanalyse 5-1
Datenbank-Connectivity 5-3
Datenbank-Desktop 5-4
Datenbanken 2-18
  Architektur von Datenbankverbindungen 3-7
  Beispiel 3-6
  Daten speichern 3-16
  Reports generieren 5-1
  Tools und Utilities 5-3
  Werte abrufen 3-5
  zugreifen auf 2-18, 3-5, 3-7, 5-3
Datenbank-Explorer 2-18, 5-4
Datenbankverbindungen 2-18
Daten-Dictionary 5-4
Datengitter 3-6, 3-7
Datenmengen 3-7
Datenmodul-Designer 2-11, 5-3
Datenmodule 2-11
Datenquellen 3-5
Datensensitive Steuerelemente 3-7, 5-3
Datensteuerelemente 3-14
DB2 5-3
dBASE 5-3
DBGrid (Komponente) 3-6, 3-7
DBMemo (Komponente) 3-14
DBText (Komponente) 3-14
Debugger 2-17
  externe Fehlersuche 2-18
  Multiprozeß-Fehlersuche 2-18
  Tool-Fenster arrangieren 4-2
Decision Support 5-1
Delphi
  Dokumentation 1-2
  Entwicklungsumgebung 2-2
  individuell anpassen 4-1
  Kunden-Support 1-4
  technische Unterstützung 1-4
  Überblick 1-1
  Versionen 5-1
DFM-Dateien 2-7
Diagramme 5-1
DLLs 5-4
Dokumentation 2-14
  Siehe auch Hilfesystem
Dynamische Link-Bibliotheken (DLLs) 5-4, 5-5

E

Editor *Siehe* Quelltexteditor
Eigenschaften
  anzeigen 3-3
  Einstellungen zur Entwurfszeit 3-3
  Werte festlegen 3-3, 3-16
Eigenschaftseditoren 3-15
Ellipsen-Schaltfläche 3-15
Entscheidungsunterstützung 5-1
Entwickler-Support 1-4
Entwurfszeiteigenschaften 3-16
  anzeigen 3-3
Ereignisbehandlungsroutinen 3-16
  anzeigen 2-6
  Standard 3-16
Ereignisse 2-6
  auswählen 2-6
  Standardereignisse 3-16, 3-17
Ereignisse (Registerkarte im Objektinspektor) 2-6
Exceptions 5-2
Experten 2-19, 4-6, 5-6
Externe Verbindungen 2-18

F

Farben 3-3
  Quelltexteditor 4-7
Fehler
  Anwendungen debuggen 2-17
  Compiler 2-15
Felder 3-6
Fenster
  andocken 4-1
Formulardateien 2-7, 3-2
Formular-Designer
  Optionen 4-7
Formulare 3-2
  Komponenten hinzufügen 2-4
  neue 4-5
  Objekte hinzufügen 3-3
  Standard 4-5
FoxPro 5-3

G

Grafiken
  hinzufügen (Beispiel) 3-12

H

Hauptfenster
  Elemente anordnen 4-4
  Elemente vom Desktop entfernen 4-4
Hilfe
  Liste der Hilfedateien 1-2
Hilfehinweise 3-5
Hilfesystem
  anpassen 4-11
  kontextsensitive Hilfe 2-14
  zugreifen auf 2-14
Hintergrundfarben 3-3

I

IDE 2-2
  Komponenten hinzufügen 4-10
Informix 5-3
Input-Method-Editoren (IMEs) 5-7
Installations-Tools 5-6
InstallShield Express 5-6
Instantiieren von Objekten 3-4
Integrated Translation Engine 5-7
Integrierte Entwicklungsumgebung (IDE) 5-1
InterBase 5-3
  InterBase Express (IBX) 5-3
ITE 5-7

J

Jahr-2000-Problem 1-4

K

Kapselung 5-2
Klassen 3-4
Klassenvervollständigung 2-16
Komponenten 2-4, 3-2, 3-3
  Siehe auch VCL
  anpassen 2-5, 5-5
  Definition 2-4
  hinzufügen 3-4
  installieren 4-10, 5-5
  schreiben 5-5
  Überblick 2-4
  VCL-Hierarchie 5-2
  von Drittanbietern 5-5
  vordefinierte 3-3
Komponentenpalette 3-3, 3-4, 5-2
  anpassen 4-8
  neu konfigurieren 4-9
  Überblick 2-4
Komponentenschablonen erstellen 4-10
Konsolenanwendungen 5-4
Kontextmenüs 2-3
Kundendienst 1-4

L

Lokale Menüs 2-3

M

MainMenu (Komponente) 3-9
Mehrschichtige Anwendungen 5-1
Mehrseitige Dialogfelder
  in der IDE konfigurieren 4-3
Memofelder erstellen 3-14
Memoobjekte 3-14
Menu (Komponente) *Siehe* MainMenu (Komponente)
Menüs 2-3, 3-2, 3-8
  hinzufügen (Beispiel) 3-9
  organisieren 4-4
Microsoft Transaction Server (MTS) 5-1
Multithreading 5-1

N

Neue Anwendung (Befehl) 3-1
Neue Einträge (Dialogfeld, Datei / Neu) 2-19, 2-20, 4-6

O

Objektablage 2-19, 4-5
Objekte 3-2
  Eigenschaften festlegen 3-16
  hinzufügen 3-3
  instantiieren 3-4
Objekthierarchie 5-2
Objektinspektor 3-3
  Eigenschaften anzeigen 3-3
  Eigenschaften festlegen 3-16
  Überblick 2-5
ODBC 5-3
OnClick (Ereignis) 3-17, 5-2
Online-Hilfe
  anpassen 4-11
  kontextsensitive Hilfe 2-14
  zugreifen auf 2-14
OpenHelp (Dienstprogramm) 4-11
Oracle 5-1, 5-3

P

Packages 5-4, 5-5
Panel (Komponente) 3-12
Paradox 5-3
Programmierhilfe 2-15, 4-7
  Ausdrücke auswerten 2-16
  Code-Parameter 2-15
  Code-Vervollständigung 2-15
  Code-Vorlagen 2-15
  Symbolinformationen 2-16
Projekt-Browser 2-10
Projektdateien 3-1
Projekte 3-1
  neue 4-5
  Standard 4-5
  Standard-Projektoptionen 4-5
Projektgruppen 2-10
Projektoptionen (Dialogfeld) 2-12, 4-5
Projektverwaltung 2-9

Q

Quelltext 3-16
  bearbeiten 2-7
  Programmierhilfe 2-15, 4-7
  Standard-Ereignisbehandlungsroutinen 3-16
  VCL 5-2
Quelltextdateien 3-1
  ändern 2-7
Quelltexteditor 2-7, 2-16, 3-16
  Einzug festlegen 4-7
  Farben festlegen 4-7
  Navigation 2-8
  Optionen 4-7
  Schriften festlegen 4-7
  Syntaxhervorhebungen 4-7
  Überblick 2-7

R

Raster (Formular-Designer)
  Ausrichtfunktion 4-7
Registerkarten
  Komponentenpalette 3-3
Reports 5-1

S

SaveDialog (Komponente) 3-16
Schaltflächen (VCL) 5-2
Schrifteditor 3-15
Schrifteinstellungen ändern 3-15
Service-Anwendungen 5-4
Speichern von Dateien 3-16
SQL 5-1
SQL Links 5-3
SQL-Explorer 2-18, 5-4
SQL-Server 5-3
Standard-Ereignisbehandlungsroutine 3-16
Standardereignisse 3-16, 3-17
Standardformulare 4-5
Standard-Hauptformular 4-6
Standardprojekte 4-5
Standard-Projektoptionen 2-12, 4-5
Statusleisten
  Hilfehinweise 3-5
  in Formular einfügen 3-5
Steuerelemente 3-2
  Datenbanken 3-7, 5-3
  datensensitive 3-7
Sybase 5-3
Symbolinformationen 2-16
Symbolleisten 2-3, 3-8
  hinzufügen (Beispiel) 3-11
  individuell anpassen 4-4
  organisieren 4-4

T

Table (Komponente) 3-7
Tafeln
  Titel entfernen 3-12
Tastenbelegung 4-7
Tastenkürzel 2-3
TComponent 5-2
Technische Unterstützung 1-4
Text
  Schriftart festlegen 3-15
  Steuerelemente 3-14
  zu Formularen hinzufügen 3-14
Textobjekte 3-14
Titel entfernen 3-12
TMenuItem (Klasse) 3-16
To-Do-Listen 2-11
Tool-Fenster andocken 4-1
Tools
  Einstellungen festlegen 4-7
Tutorial 3-1
Typbibliotheken 5-6
Typbibliotheks-Editor 5-6
Typographische Konventionen 1-4

U

Umgebungsoptionen (Dialogfeld) 2-12, 4-7
Unit-Dateien 3-2

V

VCL 5-1
  Quelltext 5-2
Versionen von Delphi 5-1
Vervollständigung von Klassen 2-16
Verzeichnisse 3-1
Vordefinierte Komponenten 3-3
Vorlagen 2-19
  Siehe auch Komponentenschablonen

W

Web-Site (Entwickler-Support) 1-4
Windows-Botschaften 5-2