

2 Erste Schritte

In diesem Kapitel werden Sie zum ersten Mal in direkten Kontakt mit Eclipse und CDT treten. Zunächst werden die Konzepte von Eclipse besprochen und allgemeine Hinweise über das Arbeiten mit der Benutzeroberfläche von Eclipse gegeben. Im zweiten Abschnitt geht es dann gleich los: Sie werden ein kleines Projekt mit Eclipse und CDT erstellen und alle essentiellen Stufen einer Implementationsphase durchlaufen. Im weiteren Verlauf werden dann wieder allgemeine Eclipse-Features besprochen. Dazu zählen das Hilfesystem, die Erweiterungsoptionen und die von der Eclipse-Plattform angebotenen Möglichkeiten, das Benutzerinterface den eigenen Bedürfnissen anzupassen.

2.1 Konzepte des Eclipse-Frameworks

Bevor wir mit dem ersten Beispiel beginnen können, werden in diesem Abschnitt zunächst einige Eclipse-spezifische Begriffe und Bedienungselemente eingeführt. Leser, denen Eclipse bereits vertraut ist, können diesen Abschnitt bedenkenlos überspringen und sich gleich Abschnitt 2.2 zuwenden, in dem wir ein erstes C-Projekt erstellen und dabei den Editor und den Debugger ein wenig kennenlernen.

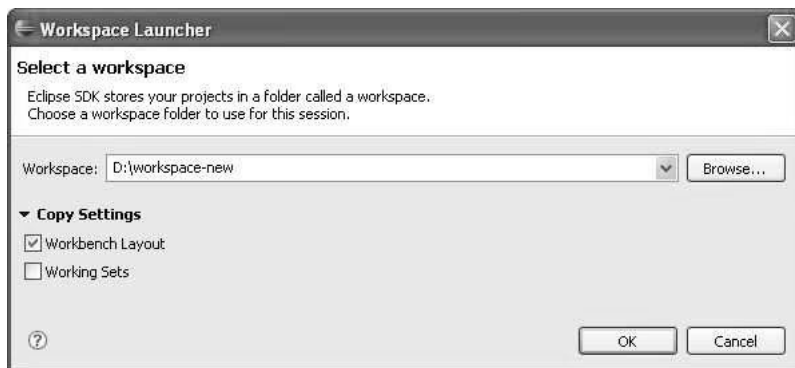
2.1.1 Workspace

Ein *Workspace*, zu Deutsch Arbeitsbereich, gehört zu jeder gestarteten Eclipse-Instanz. Er ist ein Ordner im Dateisystem, unter dem Projekte, deren Einstellungen sowie die Einstellungen der Plattform abgelegt werden. Der Zugriff auf den Workspace ist exklusiv, d.h., lediglich eine gestartete Eclipse-Instanz darf darauf zugreifen. Mehrere Eclipse-Instanzen können allerdings gestartet werden, wenn sie jeweils verschiedene Workspace-Pfade benutzen.

Ist Eclipse einmal gestartet, so können Sie im Menü *File > Switch Workspace* zu einem bereits eingerichteten Workspace umschalten. Wenn Sie wünschen, einen weiteren Workspace-Pfad einzurichten, dann wählen Sie im selben Menü den Menüpunkt *Other...* aus. Daraufhin öffnet

sich der schon bekannte Workspace-Selektor, mit dem Unterschied, dass Sie in *Copy Settings* die Einstellungen des aktuellen Workspace mit in den neuen übernehmen können.

Abb. 2-1
Der aus Eclipse
hervorgerufene
Workspace-Selektor



2.1.2 Ressourcen

Unter *Ressourcen* versteht man in der Eclipse-Terminologie all das, was auf einem Dateisystem abgelegt wird. Neben *Dateien* und *Ordern* gilt das *Projekt* als eigenständiger Ressourcentyp. Die Organisation ist hierarchisch. Projekte befinden sich auf der ersten Ebene. Projekte und Ordner werden auch als *Container* bezeichnet und können Dateien und weitere Ordner beinhalten. Dabei müssen Dateien und Verzeichnisse eines Projektes nicht zwangsläufig auch physisch innerhalb des Projektpfades abgelegt werden, sondern können sich als verknüpfte Ressource (engl. *linked resource*) auch außerhalb davon aufhalten.

2.1.3 Eclipse-Workbench

Die Arbeitsschritte des Entwicklers werden in der *Workbench* realisiert. Eine Workbench, von der durchaus mehrere Instanzen geöffnet sein können, ist ein Fenster mit einer Reihe verschiedenartiger Bereiche, deren konkrete Anzahl und Anordnung sehr frei vom Benutzer gestaltet werden kann. Abbildung 2-2 zeigt eine dem Auslieferungszustand ähnliche Aufteilung, wobei noch einige Projekte geöffnet sind. Wichtige Bedienelemente sind dort besonders hervorgehoben. Diese werden in den folgenden Abschnitten beschrieben.

2.1.4 Menüleiste

Unter der Titelleiste des Fensters befindet sich die Pulldown-Menüleiste, über die die möglichen Kommandos ausgeführt werden können, die

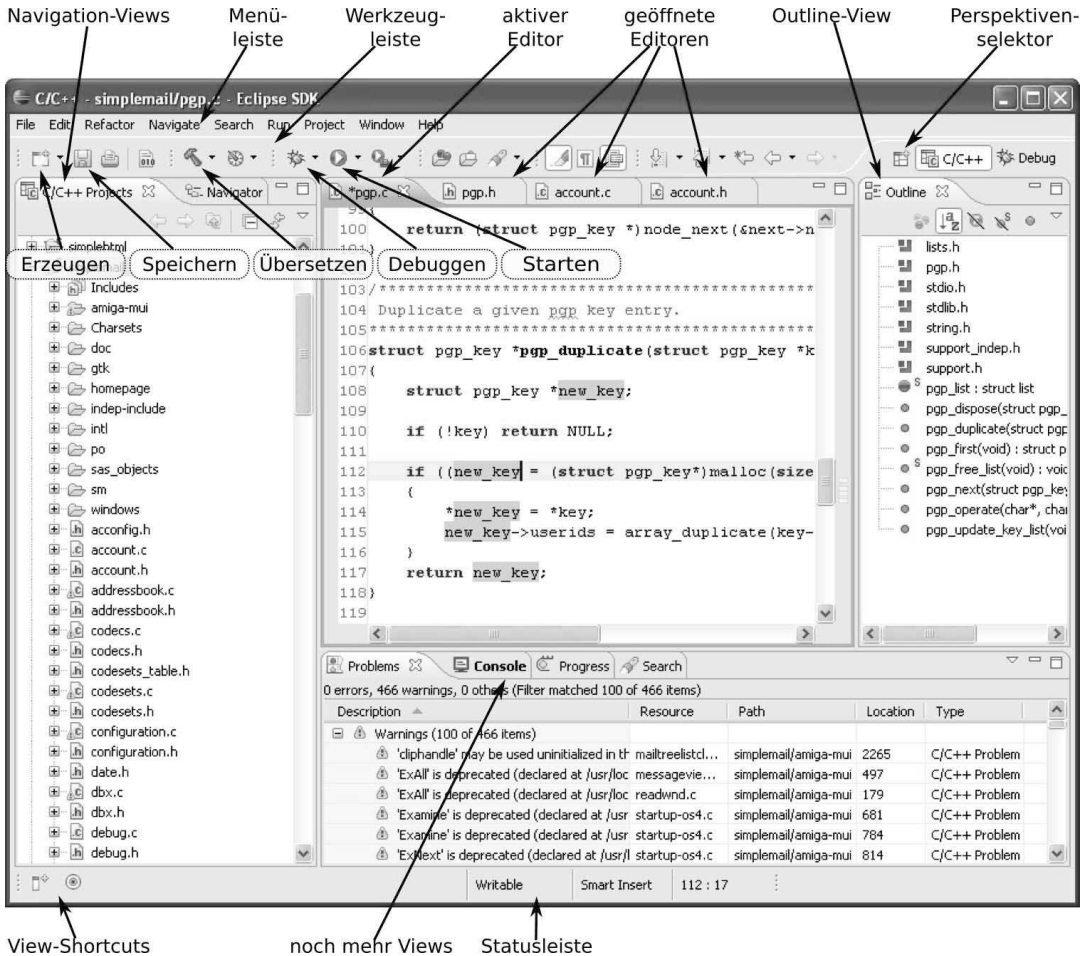


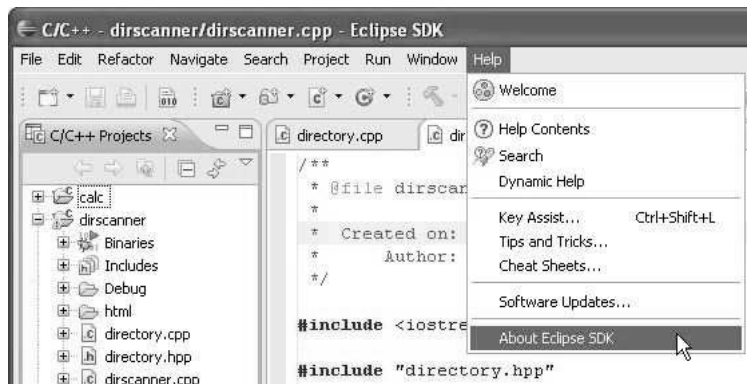
Abb. 2-2

Die Eclipse-Workbench

dann irgendeine Veränderung hervorrufen. Etwas größer ist die Menüzeile in Abbildung 2-3 dargestellt. Deren Aufbau hält sich im Grunde an die Konventionen für Benutzeroberflächen. Im *File*-Menü befinden sich Aktionen, um Dateien zu öffnen, zu sichern und so weiter. Im *Edit*-Menü befinden sich typische Aktionen, die z.B. das Kopieren und Einfügen von ausgewählten Elementen veranlassen. Hilfe erhalten Sie mit dem *Help*-Menü, das ganz rechts platziert ist und in der Abbildung aufgeklappt ist. Zwischen den genannten Menüs befinden sich Einträge, deren Aufbau oder gar Vorhandensein vom aktuellen Fokus, aber auch von den installierten Erweiterungen abhängig ist.

Abb. 2-3

Die Menüzeile, in der das Menü Help ausgewählt ist



2.1.5 Werkzeugeleiste

Unter der Menüleiste befindet sich standardmäßig die Werkzeugeleiste. Die Werkzeugeleiste beinhaltet kleine Buttons, über deren Aktion ein Symbol oder ein Tooltip Aufschluss gibt. Der Umfang der Aktionen ist gegenüber der Menüleiste auf die am häufigsten ausgeführten beschnitten, dafür können Sie diese schneller erreichen.

Die Buttons sind ihrer Funktion nach in Gruppen angeordnet, wobei Sie die Position der einzelnen Gruppen über eine kleine vertikale Leiste (engl. *trim*) innerhalb der Werkzeugeleiste verändern können, gegebenenfalls auch in weiteren Zeilen. In Abbildung 2-4 beispielsweise wurden die Gruppen in zwei Zeilen aufgeteilt. Einige Buttons der Werkzeugeleiste sind zusätzlich mit einem Dreieck versehen, das auf ein Pulldown-Menü aufmerksam macht. Mit dessen Hilfe kann die Aktion präzisiert werden. Beispielsweise bietet das Pulldown-Menü des *New C/C++ Project*-Buttons (📁) die Möglichkeit an, ein neues C-Projekt oder ein neues C++-Projekt anzulegen oder ein bereits existierendes Eclipse-Projekt in ein C/C++-Projekt zu transformieren. Auch das können Sie in der Abbildung sehen.

2.1.6 Statusleiste

Wie von anderen Programmen bekannt, befindet sich eine Statusleiste im unteren Teil des Fensters, die beispielsweise für einen aktiven Editor die aktuelle Zeilennummer angibt.

2.1.7 Editoren und Views

Editoren

Der Hauptbereich der Oberfläche wird durch *Editoren* und sogenannte *Views* beansprucht. Editoren werden, wie der Name schon sagt, hauptsächlich zum Bearbeiten von Quelltexten oder allgemeiner von Dateien

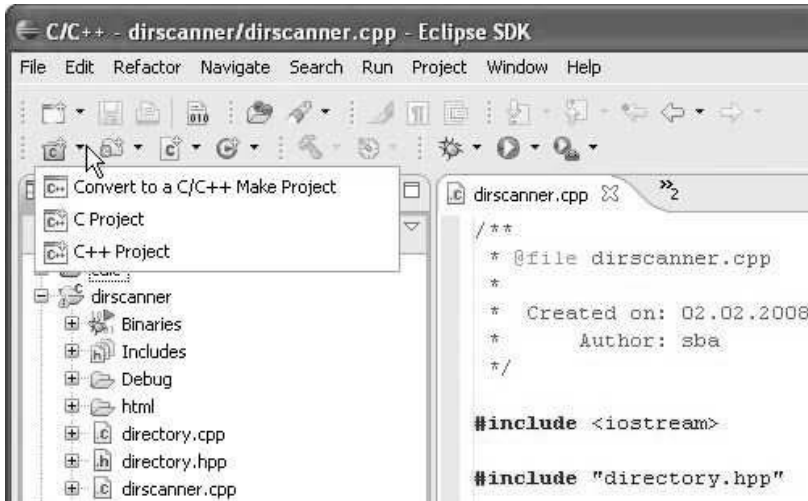


Abb. 2-4

Buttons sind nach ihrer Funktion gruppiert und können Pull-down-Menüs besitzen.

benutzt. Editoren befinden sich im Editorfeld, das in der Abbildung 2-2 deutlich die zentrale Stelle einnimmt und pro Workbench höchstens einmal vorkommt.

Demgegenüber dienen Views der Darstellung von spezifischen Informationen. Dazu zählen beispielsweise andere Darstellungsformen des gerade in Bearbeitung befindlichen Quelltextes, wie es durch *Outline-View* realisiert ist und auf der rechten Seite der Abbildung zu sehen ist. Er gibt einen Überblick über definierte Funktionen und andere Elemente. Ein View, der eine gänzlich andere Information anzeigt, ist der *Variables-View*. Ihn werden Sie vor allen Dingen beim Debugging nützlich finden. Auch die in der Abbildung links dargestellte Projekt-hierarchie ist ein View oder, genauer gesagt, ein Vertreter der *Navi-gation-Views*, die im Allgemeinen die hierarchische Organisation der Ressourcen widerspiegeln.

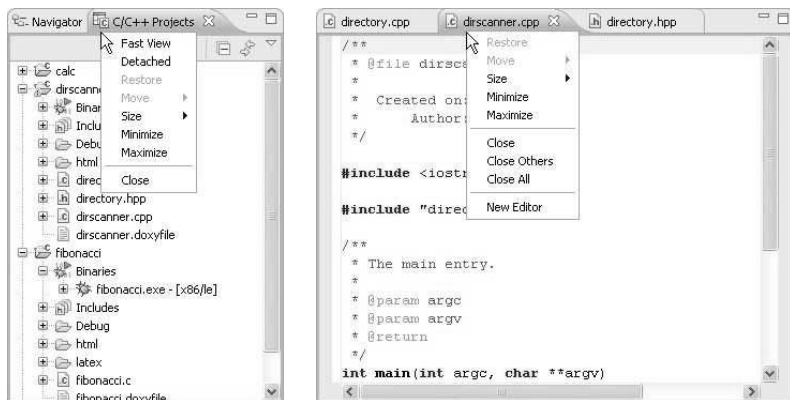
Views

Identifizierbar sind Editoren und Views immer anhand der Titelleiste, die wie die Fensterdekoration des Desktops in einem aktiven und einem inaktiven Zustand gezeichnet werden kann. Ebenso ist vom Desktop die Möglichkeit entlehnt, die Elemente per *Drag'n'Drop* über ihre Titelleiste auf der Oberfläche verschieben zu können. Editoren dürfen Sie auf diese Weise ausschließlich in ihrem Bereich neben-, unter- oder übereinander umstellen, wobei letztere Variante sicherlich die gebräuchlichste ist. Bereiche mit überdeckenden Editoren haben dann die Form eines Registerfelds, in dem die Registerkarten mit den geöffneten Editoren korrespondieren. Dagegen lassen sich Views nur außerhalb des Editorfelds platzieren, bieten Ihnen ansonsten aber mehr Möglichkeiten der Positionierung. So können sie auch von der Workbench auf

Titelleiste

den Desktop gezogen werden, was sie in gewöhnliche Fenster verwandelt.

Abb. 2-5
View und Editor in
Gegenüberstellung. Die
Kontextmenüs können
durch einen Rechtsklick
auf die Titelleiste
hervorgebracht
werden.



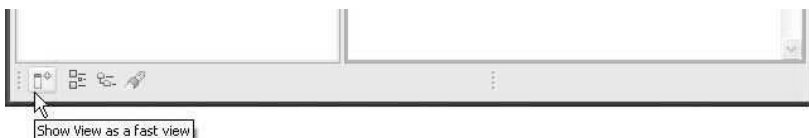
Operationen über der
Titelleiste

Abbildung 2-5 stellt einen View einem Editor gegenüber. In beiden Elementen sehen Sie ein Kontextmenü, das Sie durch einen Rechtsklick über der jeweiligen Titelleiste hervorbringen können. Darüber ist es möglich, die aus grafischen Benutzeroberflächen bekannten Operationen wie *Minimieren*, *Maximieren* und so weiter aufzurufen. Andere Auswahlmöglichkeiten beschränken sich auf nur einen Typ. So können Views zusätzlich vom Workbench-Fenster über das besagte Menü mit *Detached* als eigenständige Fenster herausgelöst werden, was dem Herausziehen des Views entspricht, oder in den Zustand *Fast View* übergehen.

Fast View

Ein *Fast View* ist eine Art ikonifizierter View, dessen Symbol in einer speziellen Werkzeugleiste erscheint, die standardmäßig im unteren Teil des Fensters ihren Platz findet. Ein ikonifizierter View wird bei Klick auf sein Symbol dargestellt, verschwindet dann aber wieder, sobald ein anderes Element aktiviert wird. Sinnvoll ist dieser Zustand für nur gelegentlich eingesehene Views. In besetzten Zustand können Sie die Leiste in Abbildung 2-6 sehen. Dort sind gerade drei Views ikonifiziert: *Outline*, *Navigator* und *Search*. Mit dem Button *Show View as a fast view*, über den der Mauszeiger gerade schwebt, können Sie die Liste ganz flink um weitere noch nicht geöffnete Views ergänzen.

Abb. 2-6
Fast View-Leiste. Der
Navigator-View ist
gerade geöffnet.



Zum Teil lassen sich die eben beschriebenen Funktionalitäten auch mittels Tastatur herbeiführen. So können Sie beispielsweise mit STRG+M den momentan aktiven View oder Editor maximieren – sehr praktisch, wenn der der Workbench zur Verfügung stehende Platz recht klein ist und Sie gerade Modifikationen am Quelltext vornehmen.

2.1.8 Perspektiven

Eclipse und CDT bieten eine Vielzahl von Views an, deren Umgang und Nutzen wir im Laufe dieses Buches detailliert kennenlernen werden. Häufig wird es so sein, dass der aktuelle Entwicklungsschritt andere Informationen benötigt als ein folgender. Beispielsweise ist beim Editieren von Quelltexten die bereits erwähnte Projekthierarchie interessant, während diese Information speziell beim Debugging weniger von Nutzen ist. Beim Debugging hingegen möchten wir die Belegung aller im Scope vorhandenen Variablen wissen. Mit dem Konzept der *Perspektiven* bietet Eclipse die Möglichkeit, zwischen Ansichten verschiedener Aufgabenbereiche schnell umzuschalten.

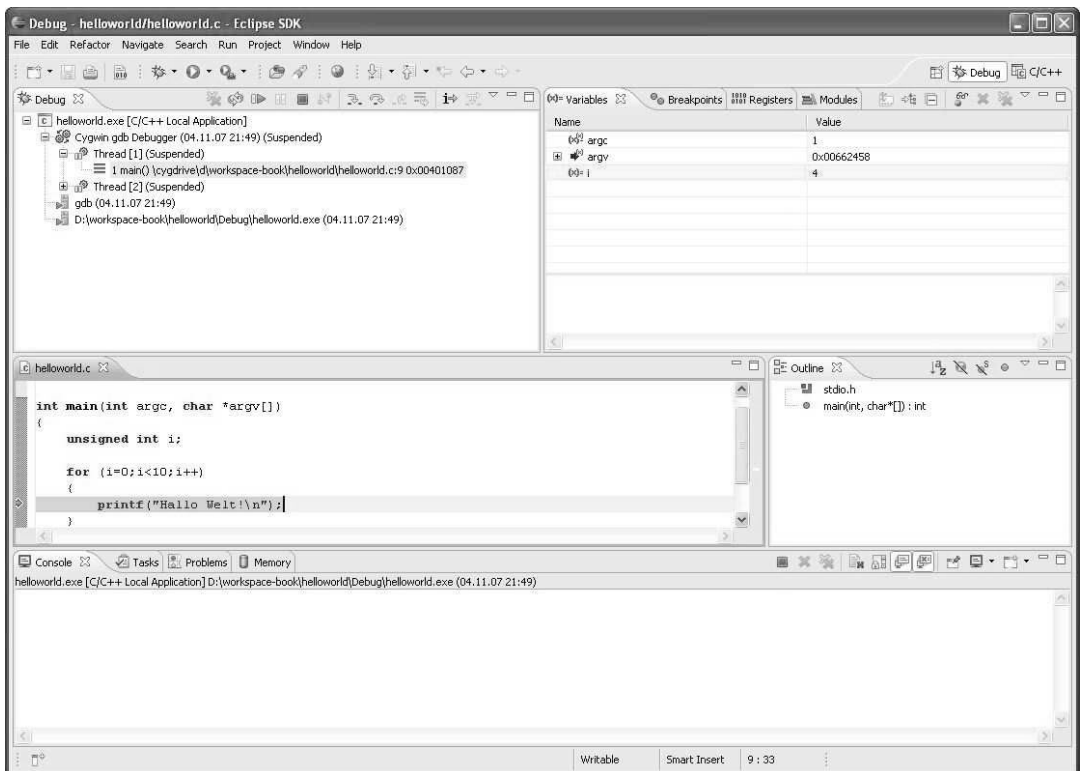


Abb. 2-7
Die Debug-Perspektive

Eine Perspektive bestimmt das Layout der Eclipse-Workbench und definiert zusätzlich mögliche Aktionen, die von Menü- und Werkzeugleiste in die Gänge gebracht werden können. Während z.B. in Abbildung 2-2 die C/C++-Perspektive gezeigt wurde, zeigt im Unterschied dazu Abbildung 2-7 die *Debug*-Perspektive, eine der Debugging-Aufgabe angepasste Benutzerschnittstelle. Geöffnet werden können Perspektiven z.B., wie in Abbildung 2-8 dargestellt, nach Auswahl des Menüpunktes *Window > Open Perspective > Other...* Daraufhin erscheint das auf der rechten Seite abgebildete Fenster, in dem alle vorhandenen Perspektiven aufgelistet sind. Bereits einmal geöffnete Perspektiven lassen sich über Shortcuts rechts oben in der Workbench abrufen. Mit dem Menüeintrag *Window > Close Perspective* können Sie die aktuelle Perspektive auch wieder schließen.

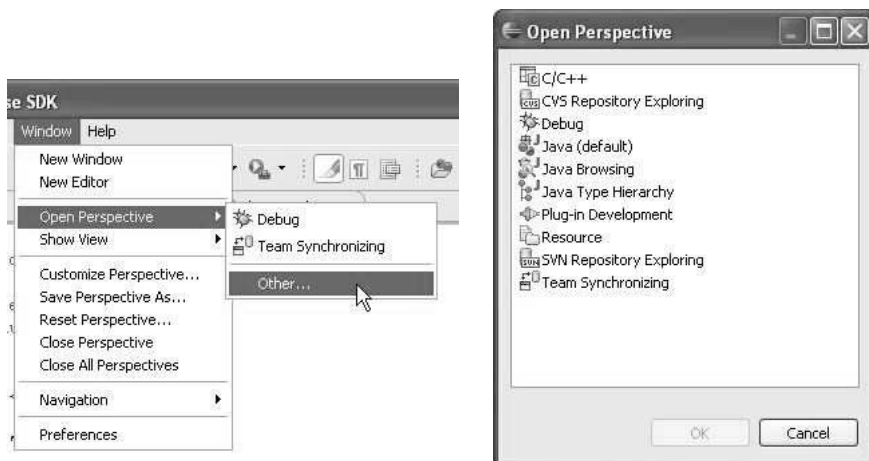


Abb. 2-8

Eine beliebige
Perspektive öffnen

Perspektiven können wahlweise in der gleichen Workbench umgeschaltet werden oder in einem neuen Workbench-Fenster erscheinen. Das Verhalten lässt sich in den Voreinstellungen, die über den Menüpunkt *Window > Preferences...* zu erreichen sind, auf der Seite *General > Perspectives* beeinflussen. Neben den durch die Plattform bereitgestellten Perspektiven ist es auch möglich, eigene Konfigurationen zu erstellen. Das aktuelle Layout lässt sich durch den Menüpunkt *Window > Save Perspective As...* unter einem neuen Namen abspeichern und mit *Window > Customize Perspective...* weiter anpassen. Insgesamt ist die Benutzeroberfläche sehr flexibel, und fast jedes Detail lässt sich ändern. Mehr darüber erfahren Sie in Abschnitt 2.6.

2.2 Das erste Projekt

Dieser Abschnitt soll Ihnen einen Vorgeschmack auf die Arbeit unter Eclipse und CDT geben, ohne auf Details oder gar spezielle Problemlösungen einzugehen. Es werden die wichtigsten Schritte anhand einer kleinen Applikation beschrieben, die die Glieder der Fibonacci-Zahlenfolge auf dem Bildschirm ausgibt. Sie werden anschließend in der Lage sein, Ihre ersten eigenen Projekte mit CDT zu erstellen und zu warten.

2.2.1 Eclipse Setup

Für die C/C++-Entwicklung empfiehlt es sich generell, das in Eclipse übliche automatische Übersetzen von Projekten, das beim Abspeichern von Quelltexten aktiv wird, abzuschalten. Im Gegensatz zum Java-Builder funktioniert der Build für C/C++-Projekte nicht inkrementell, d.h., es wird mindestens die eine Datei, an der Sie eine Änderung vorgenommen haben, komplett neu übersetzt. Dies kann je nach Größe des Projektes, der verwendeten Include-Dateien sowie der eingebundenen externen Bibliotheken erhebliche Zeit in Anspruch nehmen und damit den Arbeitsfluss beeinträchtigen. Mit Hilfe des Menüpunkt *Project > Build Automatically* können Sie diese Funktion ein- oder abschalten.

*Automatisches
Übersetzen
ausschalten*

Wenn nicht anders angegeben, finden die Interaktionen mit Eclipse in der C/C++-Perspektive statt, die von CDT für das Entwickeln von C/C++-Projekten eingerichtet wurde. Öffnen Sie diese bitte, falls sie noch nicht sichtbar ist.

*Auf die
C/C++-Perspektive
wechseln*

Jetzt stellt sich noch die Frage, woher CDT weiß, wo Compiler und die anderen Tools zu finden sind, die zu Beginn installiert wurden. CDT benutzt für das Auffinden der Programme den Kommandosuchpfad, der über die Umgebungsvariable `PATH` global definiert wird. Unter Windows wird jedoch der Compiler nicht zwingend im globalen Kommandosuchpfad zu finden sein, wohingegen dies unter Linux eher üblich ist. Sie können dies gleich einmal testen, indem Sie in einer Eingabeaufforderung das Kommando `gcc` eingeben. Wird der Befehl gefunden? Falls ja, können Sie zum nächsten Abschnitt übergehen.

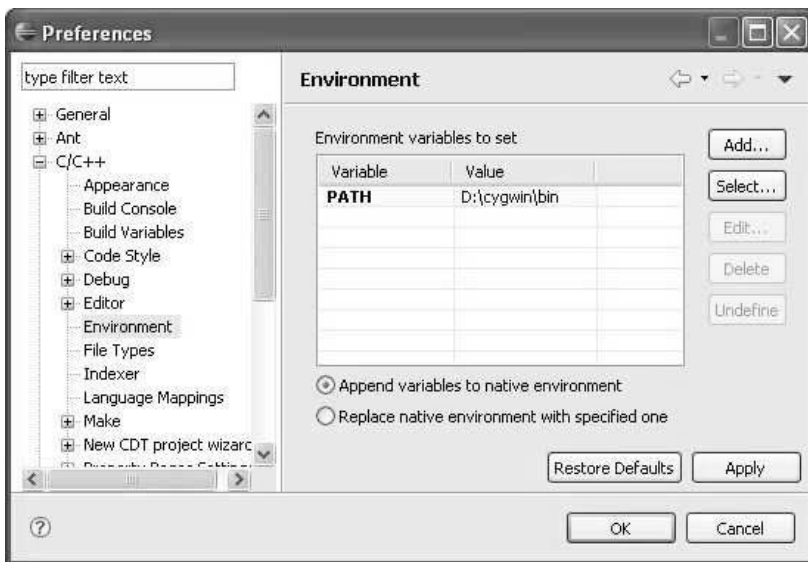
*Kann der Compiler
aufgerufen werden?*

Ansonsten müssen Sie entweder der Umgebungsvariable `PATH` den richtigen Pfad hinzufügen; hier ist gegebenenfalls ein Neustart von Eclipse erforderlich. Oder Sie gehen in den Eclipse-Voreinstellungen, die über den Menüpunkt *Window > Preferences* hervorgerufen werden, auf die Seite *C/C++ > Environment*, um die Variable ausschließlich für CDT anzupassen, wie in Abbildung 2-9 zu sehen ist. Um in der Liste den Eintrag hinzuzufügen, klicken Sie auf den Button *Add...* Im neuen Dialog geben Sie bei *Name* den Text `PATH` ein und bei *Value* das Verzeichnis, in dem sich das Programm `gcc` befindet. Verwenden Sie beispielsweise

*Kommandosuchpfad
anpassen*

se Cygwin, dann setzt sich der Eintrag aus dem Root-Verzeichnis, für das Sie sich bei der Cygwin-Installation entschieden haben, und `\bin` zusammen. Beide Fenster schließen Sie mit *Ok*.

Abb. 2-9
Die Variable `PATH` können Sie in den Einstellungen für Umgebungsvariablen anpassen.



2.2.2 Ablauf

Im Rahmen des einführenden Beispiels benutzen wir in der Werkzeugleiste die Buttons

- *New C/C++ Project*, um ein neues Projekt anzulegen,
- *New C/C++ Source File*, um eine neue Quelldatei anzulegen,
- *Save*, um Änderungen in einer Datei zu sichern,
- *Build*, um das Projekt zu übersetzen,
- *Run*, um das übersetzte Programm in Eclipse auszuführen, und
- *Debug*, um das übersetzte Programm zu debuggen.

2.2.3 Ein neues Projekt anlegen

Wählen Sie im Popup-Menü des *New C/C++ Project*-Buttons in der Werkzeugleiste den Eintrag *C Project*¹. Daraufhin öffnet sich ein Wizard, der Sie nach dem Namen des Projektes fragt. Wie in Abbildung 2-10 dargestellt, geben Sie hier bitte `fibonacci` ein. Achten Sie darauf, dass

¹Alternativ können Sie im Menü den Menüeintrag *Project > New > C Project...* auswählen.

bei *Project types* der Eintrag *Executable* hervorgehoben ist. Klicken Sie dann auf *Finish*, um die Angaben zu bestätigen und den Wizard zu schließen. Im *C/C++ Projects*-View sollte anschließend ein neues Projekt unter dem angegebenen Namen erscheinen.

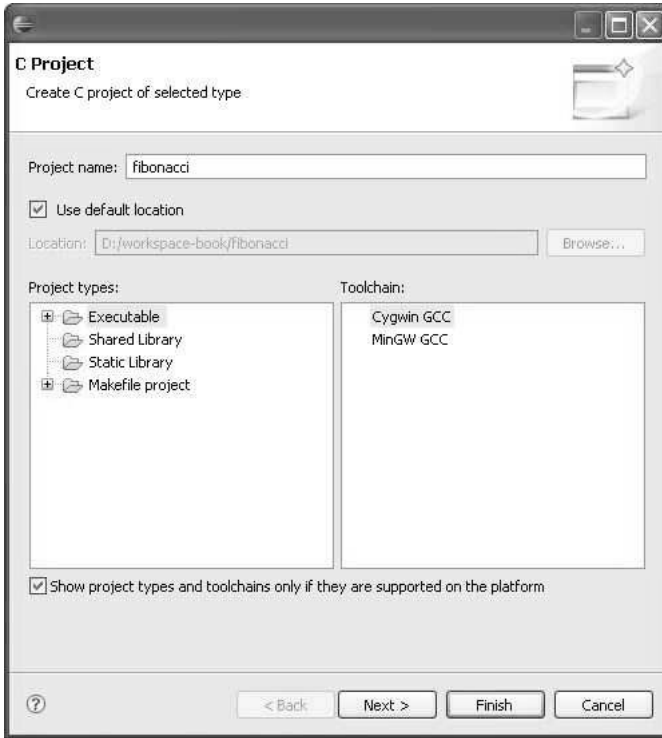



Abb. 2-10
Wizard, um neue
C-Projekte anzulegen

2.2.4 Die Quelldatei erzeugen

Selektieren Sie im *C/C++ Projects*-View unser eben erstelltes Projekt und klicken Sie in der Werkzeugleiste auf den Button  *New C/C++ Source File*, woraufhin ein Fenster erscheint, das Sie zur Eingabe des Ordners und des Namens auffordert. Stellen Sie sicher, dass bei *Source Folder* der Name unseres Projektes, also *fibonacci*, zu finden ist, und geben Sie bei *Source File* den Namen *fibonacci.c* an. Die Quelltext-Datei wird angelegt, sobald Sie die Eingabe mit *Finish* bestätigen.

Es treten nun zwei sichtbare Veränderungen innerhalb der Workbench auf. Erstens erscheint im *fibonacci*-Projekt der Dateiname *fibonacci.c*. Zweitens, und das ist viel auffälliger, wird das Editorfeld nun durch einen Editor für die neue Datei belegt, wie Sie am Titel deutlich erkennen können. Darin enthalten ist ein Cursor, der uns blinkend

zur Quelltexteingabe auffordern möchte. Seiner Aufforderung nachgehend geben Sie den folgenden Quelltext ein und behalten dabei den *Outline-View* im Auge:

```
Listing 2.1 #include <stdio.h>
fibonacci.c

unsigned int fib(unsigned int n)
{
    int i, f1=0, f2=1, f3;


    if (n<2) return n;

    for (i=2;i<=n;i++);
    {
        f3 = f1 + f2;
        f1 = f2;
        f2 = f3;
    }
    return f2;
}

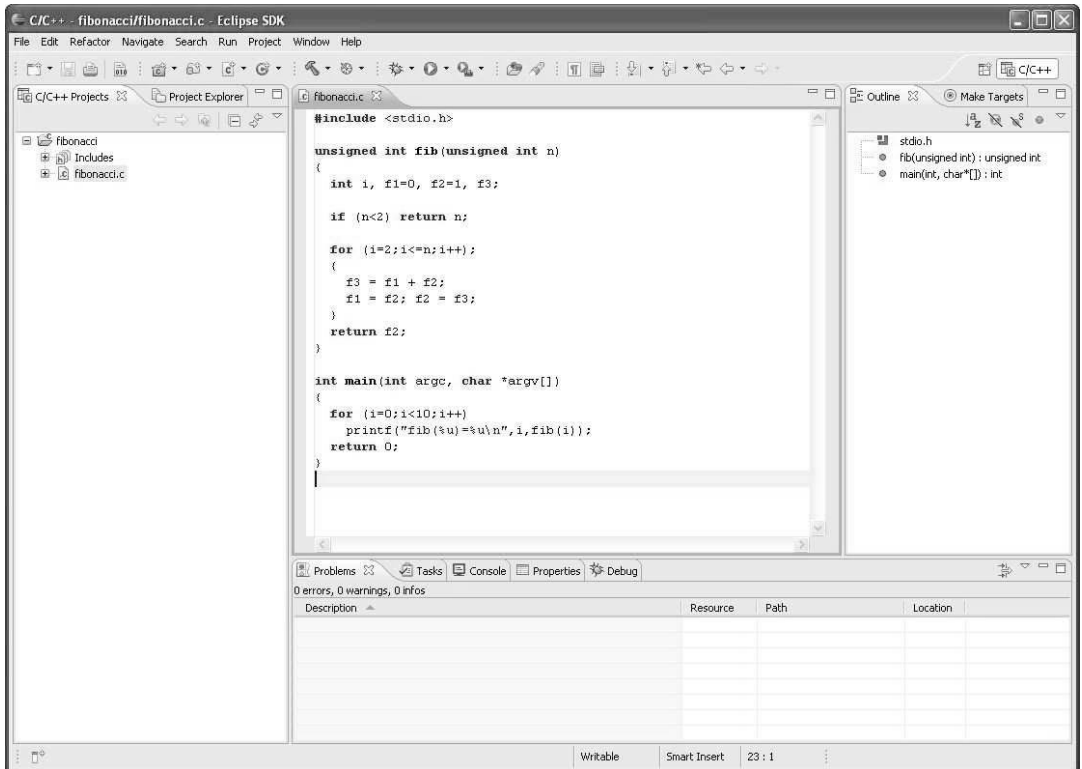
int main(int argc, char *argv[])
{
    for (i=0;i<10;i++)
        printf("fib(%u)=%u\n",i,fib(i));
    return 0;
}
```

Zwei sehr nützliche Funktionen von CDT werden Sie schon beim ersten Eingeben des Textes bemerken:

1. Sobald Sie eine kleine Pause beim Eingeben machen, wird links vom eigentlichen Texteditor an der richtigen Zeile ein Warnsymbol angezeigt, falls rein formal kein syntaktisch korrekter C/C++-Quelltext vorliegt.
2. Der *Outline-View*, der die Struktur des Programms abbildet, passt sich zügig der aktuellen Eingabe an.


Benutzen Sie den Button  Save, um die Datei zu sichern.

Sobald Sie fertig sind, betätigen Sie den Button *Save*, den Menüpunkt *File > Save* oder den Tastatur-Shortcut STRG+S, um die Datei abzuliegen. Die Benutzeroberfläche sollte sich nun in der gleichen Ansicht befinden, wie sie in Abbildung 2-11 zu sehen ist.

**Abb. 2-11**

Die C/C++-Perspektive mit geöffnetem Quelltext-Editor. Der Outline-View rechts zeigt die Struktur des Quelltextes an.

2.2.5 Übersetzen

Um das manuelle Übersetzen unseres Projektes zu starten, benutzen Sie bei aktiviertem Projekt entweder den durch ein Hämmchen symbolisierten Build-Button  aus der Werkzeugleiste oder den Menüpunkt *Project > Build Project*. Daraufhin öffnet sich ein Fenster, das Sie über den anstehenden Übersetzungsvorgang aufmerksam machen möchte. Wenn Sie den *Console-View* zur Ansicht bringen², werden Sie die aufgerufenen Compiler-Kommandos und ihre Ausgaben umgeben von ein paar CDT-spezifischen Ausgaben erkennen können.

Bestimmt haben Sie es bei der Eingabe schon bemerkt, dass sich im abgedruckten Quelltext ein Fehler eingeschlichen hat. Falls Sie diesen übernommen haben, so macht Eclipse den Fehler gleich auf verschiedene Arten deutlich, die in Abbildung 2-12 zu sehen sind.

²Wenn Sie diesen nicht finden, nutzen Sie bitte den entsprechenden Menüeintrag unter *Window > Show View* oder die Tastenkombination ALT+SHIFT+Q,C.

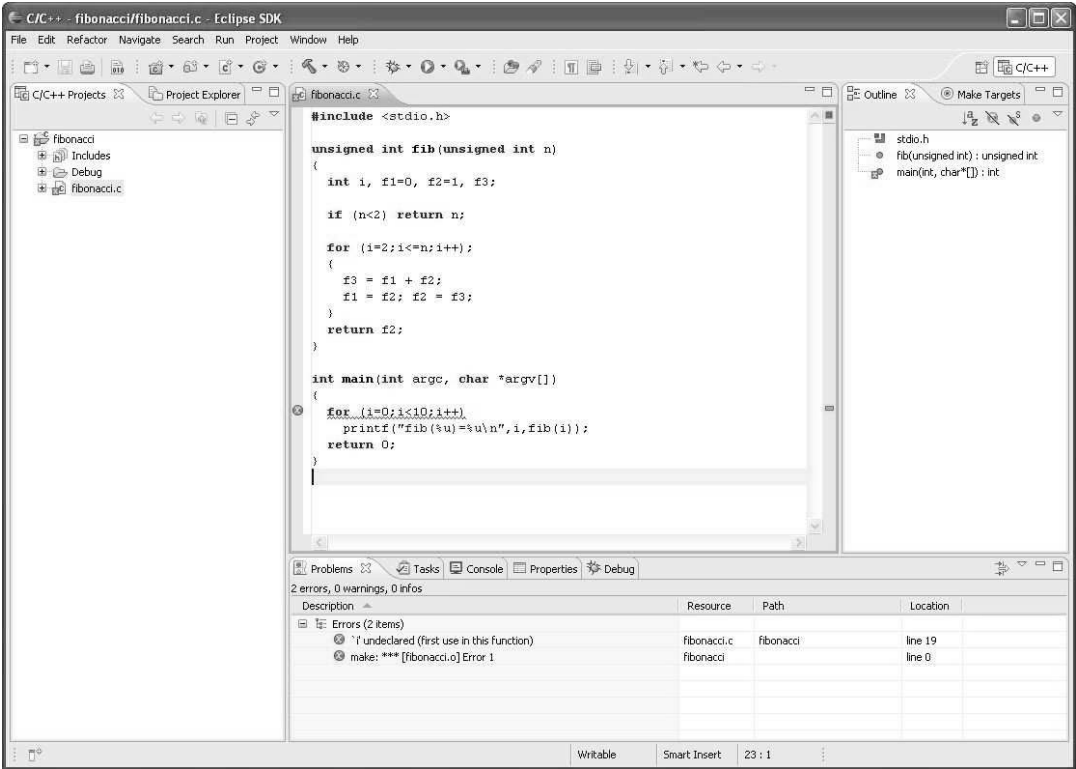


Abb. 2-12

Hinweise auf Fehler durch Marker und Unterstrichungen

Den deutlichsten Hinweis gibt dabei der Quelltexteditor selbst. Ein roter Marker auf der linken Seite des Editors zeigt die Zeile an, bei der der Compiler den Fehler vermutet. Führt man den Mauszeiger über den Marker, so erscheint eine genaue Beschreibung, die in etwa


'i' undeclared (first use in this function)

lauten sollte. Logisch, denn die Programmiersprache C erwartet die Deklaration einer Variablen vor deren Benutzung. Das haben wir nicht gemacht, und dieses Versäumnis wird zurecht vom Compiler moniert. Wir fügen also vor die Schleife ein `int i` ein, speichern den Quelltext erneut ab und lassen den Compiler wieder seine Arbeit verrichten, indem wir auf den entsprechenden Button klicken.

Dieses Mal sollten Fehlermeldungen durch den Compiler ausbleiben, die roten Marker also nicht mehr zu sehen sein. Eine weitere Änderung macht sich dann im *C/C++-Project-View* bemerkbar: Ein neuer besonders gekennzeichnete Ordner, der den Namen *Binaries* trägt, zeigt an, dass im Projekt eine ausführbare Datei vorhanden ist. Durch

Aufklappen des Ordners können Sie sich vergewissern, dass es sich tatsächlich um unsere *Fibonacci*-Applikation handelt.

2.2.6 Ausführen

Jetzt, wo alles nach Plan übersetzt wurde, wollen wir das Programm einmal starten. Stellen Sie hierfür sicher, dass im *C/C++-Projects-View* das Projekt *fibonacci* selektiert ist. Anschließend können Sie den Menüpunkt *Run > Run* oder den durch ein Play-Symbol gekennzeichneten  *Run*-Button in der Werkzeugleiste nutzen, um das Programm auszuführen, wobei implizit der Übersetzungsprozess noch einmal angestoßen wird.

Die Standardausgabe der ausgeführten Programme wird in den *Console-View* umgeleitet. In unserem Fall sollte die Ausgabe

```
fib(0)=0
fib(1)=1
fib(2)=1
...
fib(9)=1
```

ähneln, was nicht genau dem erwarteten Ergebnis entspricht. Es ist also gleich Zeit, sich dem Debugger zuzuwenden.

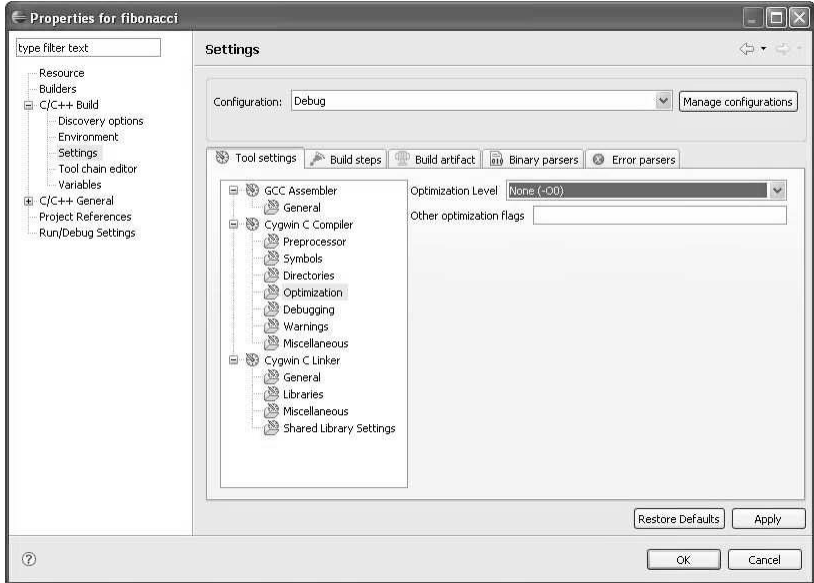
Zuvor inspizieren wir aber erneut die Konsolenausgabe. Wir können darin erkennen, dass der Compiler mit der Option *-O2* aufgerufen worden ist. Dies bedeutet, dass der Compiler den Code optimiert, was eine denkbar schlechte Voraussetzung für eine quelltextbasierte Debug-Session ist. Deshalb wollen wir zunächst die Optimierungsstufe des Compilers ändern.

Hierzu benötigen wir die *Project Properties* unseres Projektes. Sie bekommen Zugriff darauf, indem Sie im Kontextmenü des Projektordners den Eintrag *Properties* auswählen. Zugleich erscheint ein neues Fenster mit typischem Voreinsteller-Layout, d.h., links befinden sich hierarchisch angeordnete Kategorien und rechts die konkreten Einstellmöglichkeiten. Von der Menge der angebotenen Optionen soll an dieser Stelle nur die Seite *C/C++-Build > Settings* interessieren und davon ganz speziell die Auswahl *Optimization* der Registerkarte *Tool settings*; wie in Abbildung 2-13 zu sehen, setzen wir sie auf *None (-O0)*. Mit Klick auf *Ok* akzeptieren wir die vorgenommene Einstellung.

2.2.7 Programmfehler aufspüren

Sie haben das Problem sicherlich schon beim Eingeben entdeckt oder sich gar geweigert, den Quelltext in der gedruckten Form abzutippen. Eventuell haben Sie auch unbewusst den Fehler nicht mit übernommen.

Abb. 2-13
Optimierungseinstellungen. Um Source-Level-Debugging durchzuführen, ist es sinnvoll, die durch den Compiler vorgenommenen Optimierungen gänzlich abzuschalten.



Wie dem auch sei, wir werden jetzt den Debugger zu Rate ziehen, um dem kleinen Fehler, den wir bei der Konsolenausgabe bemerkt haben, auf die Schliche zu kommen.

Um das Programm im Debug-Modus zu starten, benutzen Sie den *Debug*-Button, der sich gleich neben dem *Run*-Button befindet und stilecht durch einen Käfer gekennzeichnet ist. Abermals versucht Eclipse, das Projekt zu übersetzen – dieses Mal jedoch mit der geänderten Optimierungseinstellung.

Eclipse wird Sie nun fragen, ob Sie in die *Debug*-Perspektive schalten möchten; bestätigen Sie diese Frage bitte. Wenn Sie die ebenfalls angebotene Option *Remember my decision* aktivieren, werden Sie in Zukunft gar nicht mehr gefragt. Nebenbei bemerkt können Sie diese Entscheidung in den *Run/Debug > Perspectives*-Voreinstellungen unter *Open the associated perspective when an application suspends* auch wieder rückgängig machen.

Je nach verwendetem Compiler kann es unter dem Windows-Betriebssystem vorkommen, dass Eclipse den Quelltext nicht findet, da CDT zu Beginn z.B. nichts von Cygwin-Pfaden zu wissen scheint. In dem Fall haben Sie die Möglichkeit, den Quelltext mit *Locate File...* auch für Eclipse auffindbar zu machen. Unter Linux sollten Sie derartige Probleme bei diesem Beispiel nicht haben.

Abbildung 2-14 zeigt den Aufbau der Workbench mit gestarteter Debugging-Session auf. Im Unterschied zur *C/C++*-Perspektive nimmt das Editorfeld in der standardmäßigen Einstellung deutlich weniger

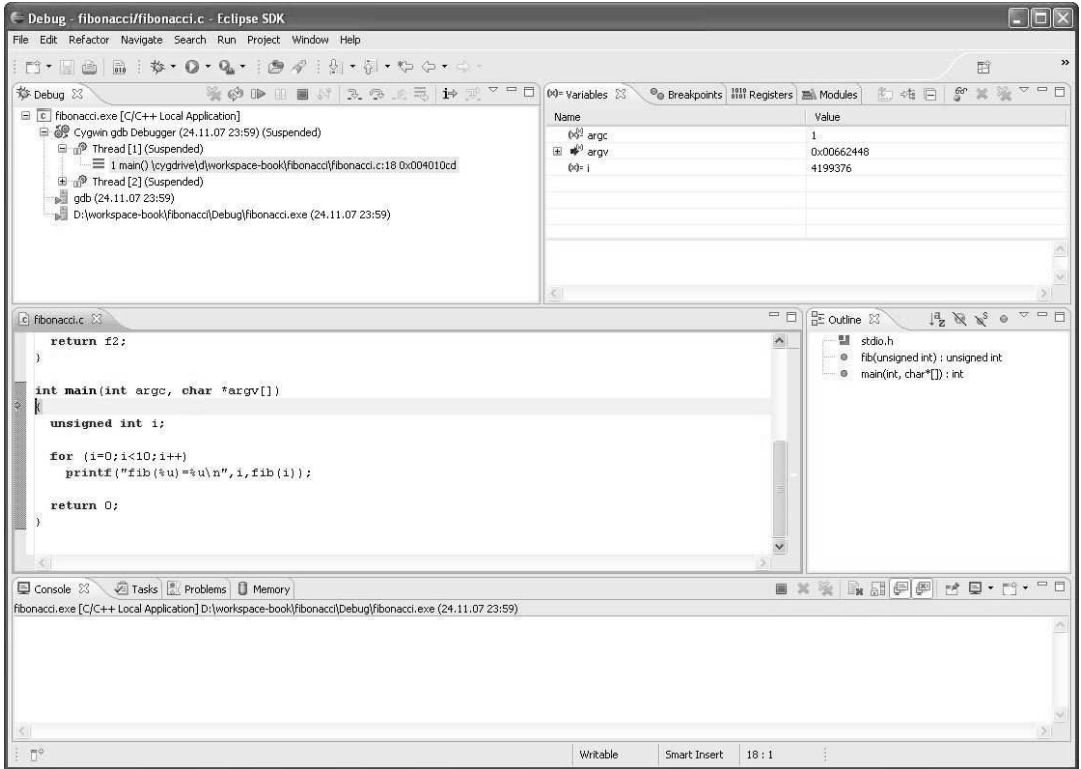



Abb. 2-14
 Erste
 Debugging-Session


Platz ein. Der C/C++-Project-View mit seiner gesamten Projekthierarchie ist gänzlich verschwunden. Der dadurch entstandene Platz wird durch zwei bisher geschlossene View-Felder ausgefüllt. Dabei ist der hier überhaupt wichtigste View der *Debugging-View*, mit dem einige Buttons assoziiert sind; durch sie kann der Ablauf des zu debuggenden Programms gesteuert werden. Im Quelltexteditor korrespondiert ein blauer Pfeil zum aktuellen Wert des Befehlszählers der CPU (im Englischen *program counter* oder *instruction pointer*), also eine Zeile mit einer Anweisung, die als Nächstes ausgeführt werden soll. Zu Beginn ist es die Zeile mit der ersten Anweisung gleich unter dem *main()*-Funktionskopf.

Um den nächsten Schritt des Programms ausführen zu lassen, benutzen wir den Button *Step Over*. Nach einem zweiten Schritt steht der Befehlszähler auf der Zeile mit dem *printf()*-Aufruf. Der *Variables-View* wurde aktualisiert und zeigt für die Variable *i* den Wert 0. Wir lassen das Programm weitere zwei Schritte laufen. Da die Ausgabe für

$i = 2$ nicht stimmte, betätigen wir anschließend den Button *Step Into* , mit dessen Hilfe wir in Funktionen hineinspringen können.

Der Befehlszähler steht nun auf der ersten Zeile der Funktion *fib()*. Der *Variables*-View wurde an die sichtbaren Variablen angepasst. Wir führen weitere Schritte aus und überprüfen dabei, ob das Ändern der Variablen und der Kontrollfluss mit unseren Erwartungen übereinstimmen. Wenn sich der Befehlszähler hinter dem Schleifenkopf befindet, spätestens aber, wenn wir den Schleifenrumpf passiert haben, stellen wir fest, dass die Schleife nicht ordnungsgemäß abgearbeitet wird. Genau genommen wird sie überhaupt nicht abgearbeitet. Bei sorgfältigem Hinschauen ist der Übeltäter schnell erkannt: ein überflüssiges Semikolon hinter der *for*-Anweisung. Ein sehr typischer Fehler, der immer wieder auch von C-Entwicklern mit mehrjähriger Erfahrung gemacht wird; der Autor schließt sich hier nicht aus.

Noch in der *Debug*-Perspektive entfernen Sie das Semikolon und klicken auf den Button mit dem roten Stoppsymbol, um die Abarbeitung des Programmes zu terminieren. Anschließend sichern Sie die eben durchgeführte Änderung, bevor das Programm wieder zur Ausführung gebracht werden kann. Ein Klick auf den *Run*-Button in der Werkzeugleiste genügt dafür. Die Ausgabe im *Console*-View sollte jetzt tatsächlich die bekannte Fibonacci-Zahlenfolge widerspiegeln.

Benutzen Sie den Button *Stop* , um die Ausführung zu terminieren.

2.2.8 Zusammenfassung und Ausblick

In diesem Abschnitt haben Sie erfahren, wie Sie mit Eclipse und CDT neue Projekte anlegen können, haben erste Erfahrungen mit dem Editor gesammelt, einen kurzen Ausschnitt über die Arbeitsweise des Build-Prozesses kennengelernt und unter Zuhilfenahme des Debuggers einen Fehler aufgespürt. Auf Grundlage der vorgestellten Arbeitsschritte könnten Sie, wenn Sie mögen, das Buch kurz beiseite legen und zunächst Eclipse auf eigenem Wege erkunden.

Natürlich liegt der Teufel auch hier im Detail. Der Funktionsumfang der IDE ist weitaus komplexer, ansonsten wäre das Buch hier bereits zu Ende, und Sie würden sich wahrscheinlich ärgern. Für den Rest des Buchs werden Schritt für Schritt die Möglichkeiten vertieft, die Eclipse, CDT und andere Plugins Ihnen in Sachen C/C++-Programmieren bieten. Um jedoch eine Vertrautheit gegenüber der Plattform zu gewinnen, bedarf es neben dem Studium auch der praktischen Anwendung. Hier können Sie die Zügel selbst in die Hand nehmen und, ausgehend von den Beispielen, das Gelernte auf die eigenen Problemstellungen transferieren. Dabei können Sie das Buch ruhig auch mal zur Seite legen, denn Hilfestellungen, die bei einem solchen Erkundungstrip vonnöten sind, vermag Ihnen auch Eclipse in verschiedenen

Formen selbst zu geben. Auf die unterschiedlichen Facetten des Hilfesystems soll deswegen im folgenden Kapitel eingegangen werden.

2.3 Das Eclipse-Hilfesystem

Die Eclipse-Plattform kommt mit einem zentralen Hilfesystem daher. Plugins, die einer Anleitung bedürfen, können sich in das System einlinken und Ihnen ihre eigenen Inhalte anbieten. Auf diese Weise haben Sie zudem den kompletten Überblick über alle auf Ihrem System eingerichteten Plugins.

In diesem Abschnitt werden Sie erfahren, in welchen Situationen Sie auf welche Weise die Hilfe aufrufen können.

2.3.1 Dokumentationsaufbau

Die oberste Ebene der Dokumentationshierarchie besteht aus den Plugins, die eine Dokumentation beisteuern. Darunter sind dann die einzelnen Punkte gegliedert. Die Gliederungsstruktur ist den Entwicklern des Plugins prinzipiell freigestellt. Die Benutzer ansprechenden Dokumentationen haben jedoch häufig eine ähnliche Form, so dass Sie sich in neuen Beschreibungen schnell zurechtfinden sollten. Die folgenden Themen sind nahezu in jeder Dokumentation enthalten:

- *Getting started*. Hier wird der Benutzer schnell in die Arbeitsschritte für das Plugin eingeführt, häufig in Form von kleinen Tutorials.
- *Concepts*. Diese Sektionen enthalten oft eine Übersicht über das Plugin und klären über Konzepte, Begriffe und ihre Zusammenhänge auf.
- *Tasks*. Hier werden konkrete Abläufe beschrieben.
- *References*. Das ist die Referenz zum Plugin. Voreinstellungen sowie die Funktionsweisen von Editoren und Views werden erläutert.

Daneben gibt es noch Themen, zu denen keine Unterthemen existieren. Diese sind häufig:

- *Tips and tricks*, in denen sich problemorientierte Tipps finden,
- *What's new*, worin die Änderungen von einer vorherigen Version des Plugins beschrieben sind, sowie
- *Legal*, die Informationen bzgl. des Urheberrechts der Dokumentation beinhalten.

2.3.2 Hilfefenster

Zugriff auf das Hilfesystem erhalten Sie, indem Sie den Eintrag *Help > Help Contents* aus dem Menü auswählen. Eclipse zeigt darin ein zweigeteiltes Fenster, das Sie in Abbildung 2-15 sehen können. Als eines der wenigen Fenster in Eclipse ist dieses nicht modal, d.h., Sie können in anderen Eclipse-Fenstern weiterarbeiten, während die Hilfe dargestellt wird, was sicherlich den Zweck einer Hilfe sehr begünstigt.

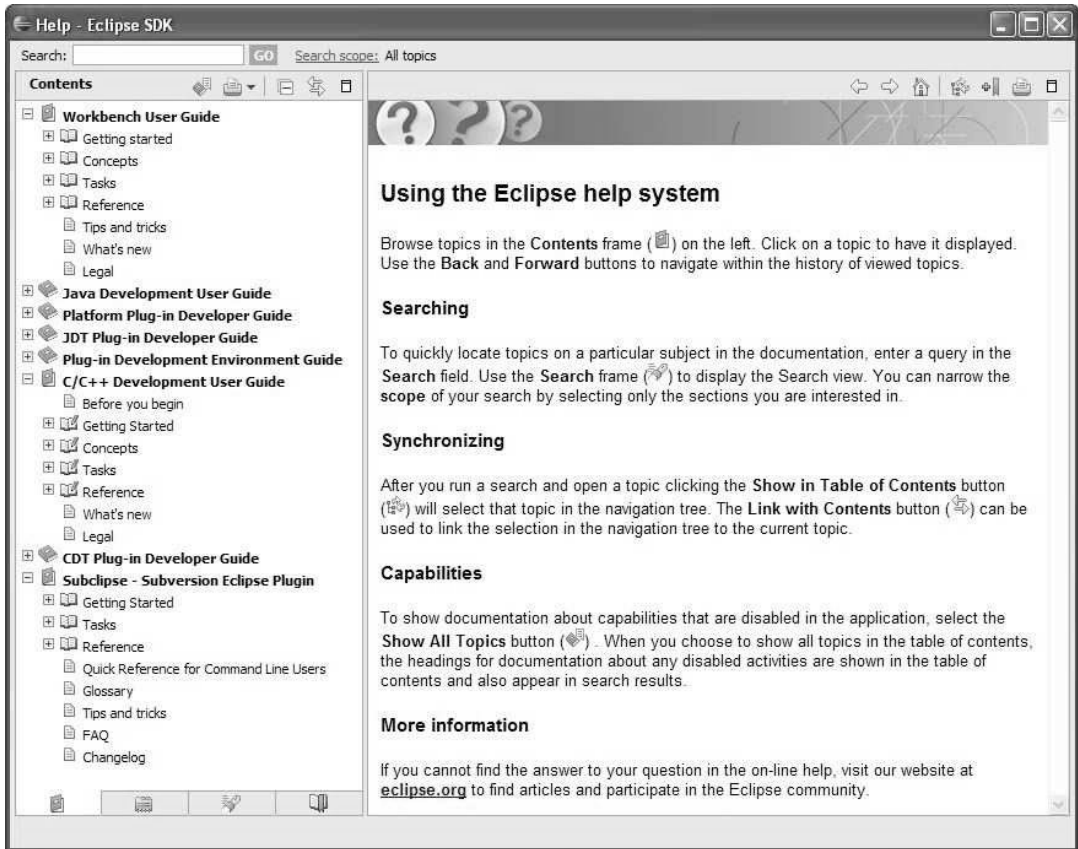


Abb. 2-15

Das Hilfefenster hat einen Aufbau, der typisch für ein hierarchisches Hilfesystem ist: links die baumartige Struktur, rechts der Inhalt.

Auf der linken Seite ist die Struktur der Hilfe zu sehen. Durch Aufklappen von einzelnen Punkten können Sie durch die Dokumentation baumartig navigieren. Der eigentliche Inhalt wird dann im großen Bereich Webbrowsers-ähnlich auf der rechten Seite präsentiert, sobald Sie ein Thema selektiert haben.

Weitere Interaktionsmöglichkeiten bieten die Buttons, die sich oberhalb der beschriebenen Bereiche befinden. Für die Gliederungsübersicht wären dies

*Aktionen der
Gliederungsübersicht*

- *Show All Topics.* Normalerweise werden in die Gliederung nur diejenigen Themen von Plugins aufgenommen, deren Komponenten auch aktiviert sind (siehe Abschnitt 2.5.8). Wenn Sie diese Option aktivieren, so fließen tatsächlich alle möglichen Themen in die Gliederung mit ein.
- *Print Topics.* Hiermit können Sie die Themen ausdrucken. Sie haben die Wahl, das Thema mit oder ohne allen zugehörigen Unterthemen auszudrucken.
- *Collapse All.* Hiermit werden die geöffneten Gliederungspunkte geschlossen.
- *Link with Contents.* Falls Sie im rechten Teil durch die Benutzung von Hyperlinks zu anderen Themen navigieren, so wird das entsprechende Thema auch in der Übersicht sichtbar gemacht, falls diese Option aktiviert ist.

Für den Fließtext stehen Ihnen neben den Browser-üblichen Navigations-Buttons *Forward*, *Backward* und *Home* noch weitere zur Verfügung:

*Aktionen über den
Fließtext*

- *Show in Table of Contents*, um die aktuelle Seite im Gliederungsverzeichnis wiederzufinden,
- *Print Page*, um die dargestellte Seite zu drucken, sowie
- *Bookmark Document*, um die aktuelle Seite zu den Hilfe-Fenster-eigenen Lesezeichen hinzuzufügen.

Den Einblick auf die Lesezeichen finden Sie im Register *Bookmarks*, die Sie im linken unteren Bereich anwählen können. Ein Klick darauf lässt die einfache Liste erscheinen, deren Elemente mit darüber gesetzten Buttons gelöscht werden können.

*Zu den Lesezeichen
finden*

Zusätzlich zu den Navigationsmöglichkeiten können Sie auch nach Begriffen suchen. Hierfür existiert ein Texteingabefeld im oberen Teil des Fensters, in dem Sie auch die Joker-Zeichen »*« und »?«, die booleschen Ausdrücke »AND«, »OR« und »NOT« sowie Anführungszeichen als Kennzeichen für zusammenhängenden Text benutzen können. Die Ergebnisse werden dann in einem weiteren Register im linken Bereich präsentiert.

Nach Begriffen suchen

Die Dokumentation gibt's auch im Netz.

Im Übrigen können Sie die Hilfe aller Eclipse-Projekte auch über das »Weltweite Netz« unter der Adresse


<http://help.eclipse.org>

abrufen. Das Benutzerinterface der Webseite ist dem hier vorgestellten ganz ähnlich. Die dort angebotene Dokumentation umfasst jedoch die Texte aus dem gesamten Spektrum der harmonisiert veröffentlichten Eclipse-Plugins des aktuellen Zyklus.

2.3.3 Kontextbezogene Hilfe

Neben der eben erläuterten statischen Hilfe, mit der Sie selbst gezielt nach Informationen suchen können, kennt Eclipse noch das Konzept der *dynamischen* oder *kontextbezogenen* Hilfe, die Sie für spezifische, gerade in Betrachtung befindliche Elemente erhalten.

Die kontextbezogene Hilfe rufen Sie auf, indem Sie den Menüeintrag *Help > Dynamic Help* auswählen oder F1 auf Ihrer Tastatur betätigen. Daraufhin öffnet sich das in Abbildung 2-16 illustrierte *Help-View*, falls es nicht schon zuvor geöffnet war.

Nutzen Sie das Fragezeichen , um kontextbezogene Hilfe in Fenstern zu erhalten.

Diese Art der Hilfe ist nicht nur für das Fenster der Workbench verfügbar, sondern auch für so manch andere Fenster. Signal hierfür ist ein kleines Fragezeichen, das in den meisten Fällen im unteren linken Teil des Fensters zu finden ist. Das Betätigen dieses Symbols oder der F1-Taste lässt dann im rechten Teil die aus dem *Help-View* bekannte Ansicht erscheinen.

Über die eingebetteten Hilfefunktionen haben Sie natürlich auch auf die aus dem Hilfe-Fenster bekannte Funktionen Zugriff. Hierfür dienen die Verknüpfungen, die sich unterhalb der *Go To*-Aufschrift befinden. Die Themen der statischen Hilfe erhalten Sie mit *All Topics*, die Suchfunktion mit *Search*, den Index über *Index* und die Lesezeichen mit *Bookmarks*. Mit dem Punkt *Relatic Topics* schalten Sie die kontextbezogene Hilfe wieder ein. Des Weiteren existieren die üblichen Navigationsbuttons: Mittels *Back*- und *Forward*-Buttons können Sie entsprechend dem Verlauf vor- oder zurückbewegen.

Hilfesystem durchsuchen

Die erwähnte Suchfunktion, die Sie auch über den Menüeintrag *Help > Search* erreichen, bietet Ihnen zur Suche gegenüber dem Hilfesystem ein weiteres Feature: Denn neben der Suche innerhalb aller lokalen Hilfetexte können Sie auch in anderen Medien suchen. Vorgabemäßig wird beispielsweise auch das WWW durch die Suchmaschine Google nach dem Begriff durchsucht. Kontrolle über dieses Feature haben Sie über den *Search scope*.

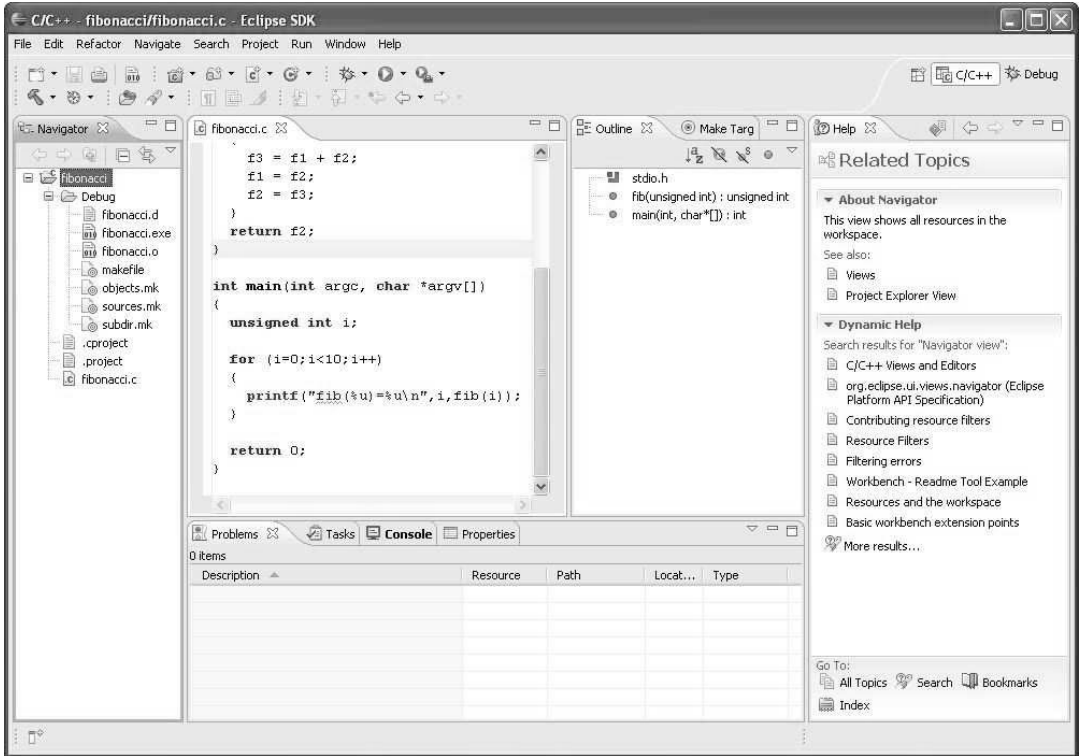


Abb. 2-16

Kontextbezogene Hilfe. Der im hier rechts befindlichen Help-View angezeigte Hilfetext richtet sich nach dem aktiven Editor oder View.

2.4 Eclipse über die Tastatur steuern

Alle Aktionen sind in Eclipse über die Tastatur abrufbar. Dies trägt zum einen der Barrierefreiheit Rechnung, zum anderen sind häufig benutzte Funktionen über Tastaturkürzel schneller zu erreichen.

Zugriff auf die Menüeinträge erhalten Sie mit F10 (oder über die linke ALT-Taste unter Windows). Sie können durch das Menü mit den Cursortasten navigieren und den Eintrag mittels RETURN bestätigen. Oder Sie wählen die Einträge mittels des in seiner Beschriftung unterstrichenen Buchstabens in Zusammenhang mit ALT aus. Das funktioniert auch für alle anderen Bedienelemente. Weitere wichtige Tastenkürzel, die die allgemeine Bedienung von Eclipse betreffen, werden in Tabelle 2-1 dargestellt.

Eine Liste aller möglichen Tastaturkürzel können Sie über den Menüeintrag *Help > Key Assist...* (STRG+SHIFT+L) einsehen. Die dort aufgelisteten Einträge sind kontextbezogen: Nur die Kommandos, die

Tab. 2-1

Wichtige Tastenkürzel

Kürzel	Funktion
Strg+F6	Editoren durchwandern
Strg+F7	Views durchwandern
Strg+F8	Perspektiven durchwandern
F10	Menüleiste aktivieren
Shift+F10	Kontextmenü für den aktuellen View
Ctrl+F10	Pulldown-Menü für den aktuellen View
F12	Editor aktivieren
Strg+M	Aktuellen View/Editor maxi-/minimieren

momentan absetzbar sind, erscheinen dort. Das funktioniert sowohl für Views und Editoren als auch für Eingabefelder in anderen Fenstern.

Die Tastaturkürzel sind nicht fest und können den eigenen Bedürfnissen angepasst werden. Eine Einführung hierfür wird noch in diesem Kapitel in Abschnitt 2.6.3 gegeben. Die in diesem Buch verwendeten Tastaturkürzel entsprechen allerdings den Vorgabewerten.

2.5 Nützliche Einstellungen

Dieser Abschnitt gibt eine Einführung in das Voreinstellungssystem von Eclipse. Eclipse bietet hier eine besonders große Fülle an Möglichkeiten. Deshalb soll Ihnen zuerst das Konzept nähergebracht werden und lediglich anhand von wenigen, aber nützlichen Einstellungen die Bedienung verdeutlicht werden. Einstellungen, die die Individualisierung der Benutzerschnittstelle von Eclipse zum Gegenstand haben, werden dann im nächsten Abschnitt behandelt. Darüber hinausgehende, wichtige Voreinstellungen, insbesondere jene, die die Abläufe in der Programmierung betreffen, werden in den einzelnen Kapiteln an entsprechender Stelle besprochen. Ansonsten gilt auch hier das Motto »Anschauen und Ausprobieren«.

2.5.1 Voreinstellungsdialog

Die Voreinstellungen rufen Sie herbei, indem Sie den Menüpunkt *Window > Preferences...* auswählen. Daraufhin erscheint ein Fenster, das so ähnlich aussehen sollte wie das in Abbildung 2-17 dargestellte. Links befindet sich eine Auswahl von Kategorien, während Sie im rechten Teil die zur selektierten Kategorie passenden Voreinstellungen vornehmen

können. Mit Hilfe der Pfeile im oberen Bereich des Fensters können durch bereits besuchte Kategorien navigieren.

Mitunter suchen Sie nach einer ganz bestimmten Kategorie. Hierfür können Sie im Texteingabefeld, das sich über der Kategorieauswahl befindet, Titel sowie Schlüsselwörter eingeben, nach denen die dargebotenen Kategorien gefiltert werden. Wenn Sie beispielsweise Einstellungen des Proxys verändern möchten, so geben Sie dort einfach proxy ein, so dass die Auswahl von Kategorien erheblich schrumpft.

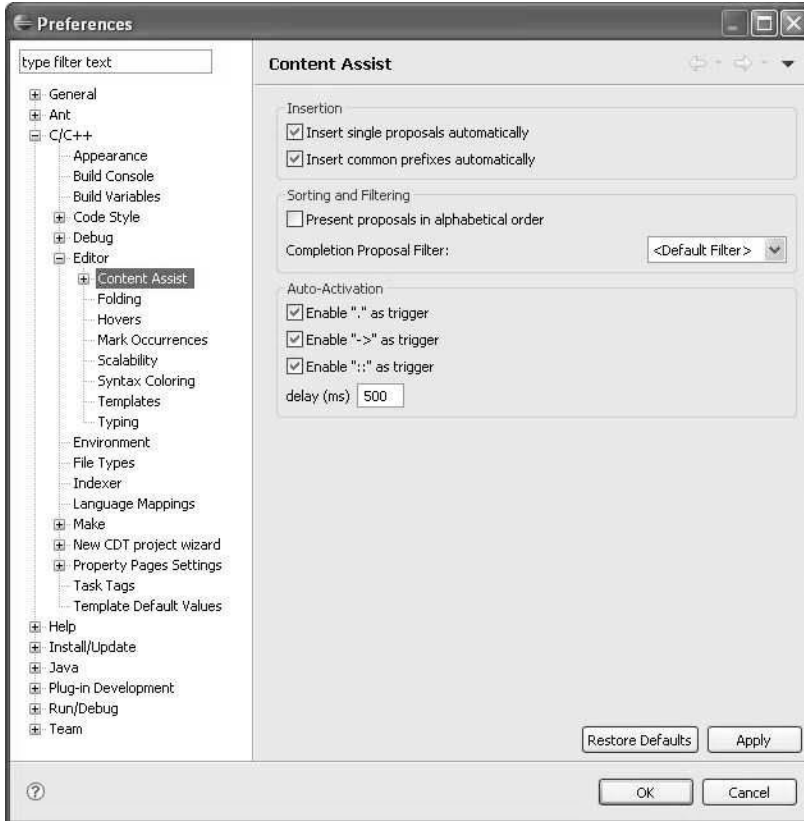


Abb. 2-17
Voreinstellungsfenster

2.5.2 Hierarchie der Voreinstellungen

Wie so alles in Eclipse ist auch die Struktur des Aufbaus der Voreinstellungen alles andere als linear. Vielmehr sind Kategorien hierarchisch angeordnet, um ein leichteres Auffinden von Einstellungen zu ermöglichen. Des Weiteren können sich auf diese Weise Plugins an beliebiger Ebene einklinken und dort neue Kategorien erstellen. Folgende Oberka-

tegorien werden im Voreinstellungsdialog bei einer Standardinstallation für C/C++-Entwicklung präsentiert.

- *General*. Hier finden Sie alle Einstellungen, die das allgemeine Verhalten und Aussehen der Workbench von Eclipse betreffen.
- *C/C++*. Diese Voreinstellungen werden durch das CDT-Plugin hinzugefügt. Hier finden sich alle Einstellungen wieder, die speziell die C/C++-Entwicklung betreffen.
- *Help*. Auch das Verhalten der Hilfe kann mit Einstellungen beeinflusst werden.
- *Install/Update*. Hier können Voreinstellungen bezüglich des Installierens von weiteren Plugins und der Strategie ihrer Aktualisierung vorgenommen werden.
- *Run/Debug*. Das Starten und Debuggen von Programmen stellt in Eclipse ein separates Konzept dar. Hier können Sie allgemeine Einstellungen vornehmen, die größtenteils visueller Natur sind.
- *Team*. Hier befinden sich die Einstellungen, die sich um die Versionsverwaltung drehen. Die Versionsverwaltung spielt eine besondere Rolle, wenn Projekte innerhalb eines größeren Teams entwickelt werden. Sie wird in aller Ausführlichkeit in Kapitel 6 behandelt.

Daneben gibt es noch weitere Kategorien. Da Sie mit dem vollständigen *Eclipse SDK* Java- sowie Plugin-Entwicklung betreiben können, finden sich dort beispielsweise zusätzlich die Kategorien *Ant*, *Java* und *Plugin Development* wieder, die jedoch allesamt weniger interessant für die C-/C++-Entwicklung sind.

2.5.3 Allgemeine Workspace-Einstellungen

Die Einstellungen, die das Verhalten des Workspace beeinflussen, befinden sich auf der Seite *General > Workspace*. Diese ist in Abbildung 2-18 dargestellt.

Automatisch bauen

Eine Philosophie von Eclipse ist, dass ein sich in Entwicklung befindliches Projekt immer in kompilierter Fassung vorliegen soll. Hierfür verantwortlich ist die Option *Build automatically*, die standardmäßig aktiviert ist und das Bauen nach jeder auf dem Datenträger gespeicherten Änderung veranlasst. Auf diese Weise werden die Turnaround-Zeiten der Entwicklung verkürzt, da der Programmierer schon bei kleinen Modifikationen sofort Feedback über eventuelle Übersetzungsfehler bekommt. Leider funktioniert dies nur vernünftig, wenn das Übersetzen an sich flott vonstatten geht.

*Ressourcen
automatisch
auffrischen*

Werden Dateien außerhalb von Eclipse modifiziert, so müssen standardmäßig die geänderten Ressourcen oder ihre Container manuell aufgefrischt werden. Zwar weist Eclipse bei der Prozessierung einer nicht

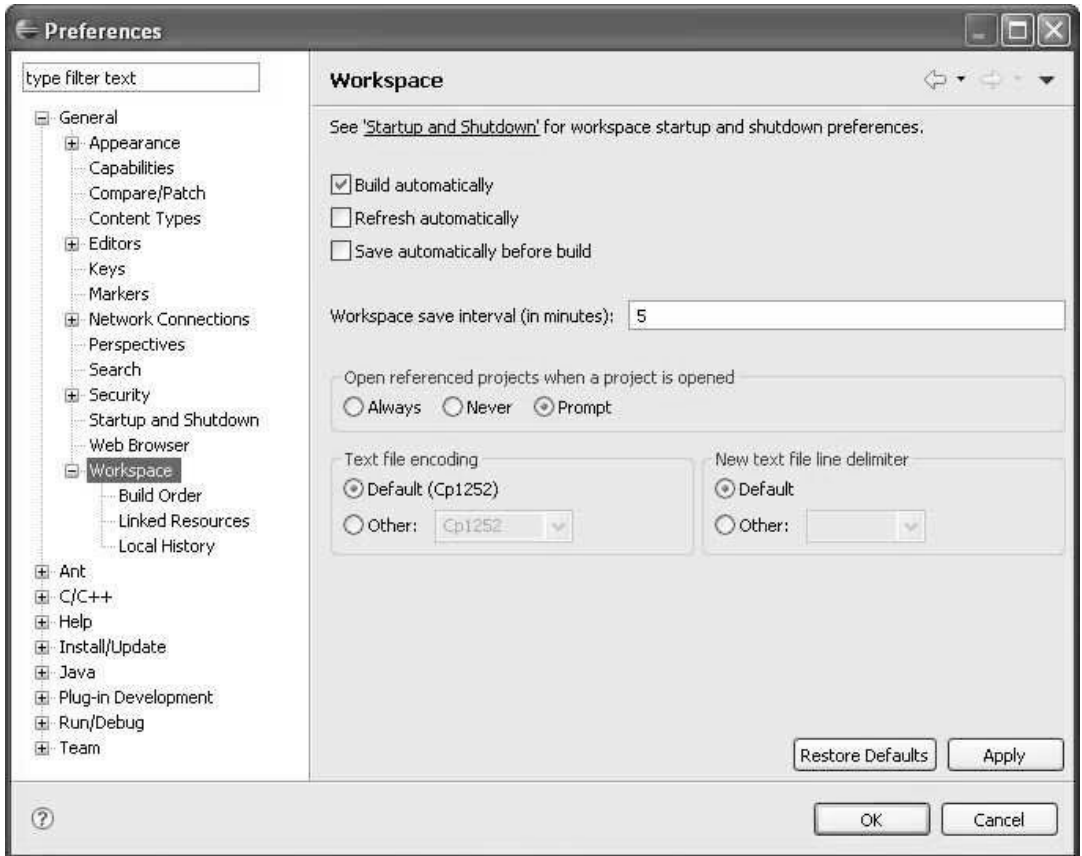


Abb. 2-18
Ansicht der
Voreinstellungen des
Workspace

unter Eclipse veränderten Ressource auf diesen Zustand hin, doch kann das anschließend notwendige manuelle Auffrischen den Arbeitsfluss stören. Abhilfe schafft hier das Aktivieren der Option *Refresh automatically*. Allerdings könnte dies zu einer allgemeinen Beeinträchtigung der Workbench-Performanz führen, insbesondere dann, wenn sich viele Ressourcen im Workspace befinden.

Mit der Option *Save automatically before build* können Sie in Eclipse veranlassen, dass die Änderungen aller Ressourcen, die von Ihnen gerade bearbeitet werden, vor dem Bauen eines Projekts auch auf den Datenträger geschrieben werden.

Weiter unten spezifizieren Sie in *Workbench save interval* die Zeit in Minuten, nach der der momentane Zustand des Workspace gesichert wird.

Automatisch speichern

Periodisch speichern

Referenzierte Projekte

Innerhalb eines Workspace können Projekte andere Projekte referenzieren, was zum Beispiel sinnvoll ist, wenn Header-Dateien oder Bibliotheken geteilt werden. Des Weiteren können Sie Projekte schließen, wenn Sie von Ihnen gerade nicht bearbeitet werden müssen. Wenn Sie ein geschlossenes Projekt erneut öffnen, können im Kasten *Open referenced projects when a project is opened* verschiedene Verhaltensweisen angegeben werden. Referenzierte Projekte werden

- *geöffnet*, wenn Sie *Always* angeben, oder
- *nicht geöffnet*, wenn Sie *Never* selektieren. Alternativ werden Sie bei jedem Öffnen
- *gefragt*, wenn Sie *Prompt* angeben.

Das gleiche Schema finden Sie auch bei anderen Voreinstellungen wieder, etwa auf der Seite *Run/Debug > Launching*.

Textkodierungen

In den beiden untersten Kästen geht es um die Kodierung der auf Datenträger gesicherten Texte. Während in *Text file encoding* die Kodierung für sichtbaren Text spezifiziert wird, wird durch *New text file line delimiter* ausschließlich die Kodierung für den Zeilenabgrenzer angegeben. Beide Einstellungen haben keine Auswirkungen auf bereits vorhandene Texte, so dass sie eigentlich nur bei einem neuen Workspace modifiziert werden sollten, wenn der Bedarf überhaupt besteht.

2.5.4 Starten und Beenden

Attribute, die den Start und das Beenden von Eclipse zum Gegenstand haben, können Sie auf der Seite *General > Startup and Shutdown* verändern. Das Interface wird in Abbildung 2-19 gezeigt.

- *Prompt for workspace on startup*. Wenn diese Option aktiv ist, erscheint beim Start von Eclipse der Workspace-Selektor.
- *Refresh workspace on startup*. Schalten Sie diese Option ein, wenn Sie wünschen, dass der Workspace bei jedem Programmstart aufgefrischt wird.
- *Confirm exit when closing last window*. Bei aktivierter Option müssen Sie das Beenden von Eclipse bestätigen, das beim Schließen des letzten Fensters erfolgen würde.

In der sich darunter befindlichen Liste *Plug-ins activated on startup* werden Plug-ins aufgezählt, die beim Start von Eclipse gestartet werden.

2.5.5 Netzwerkeinstellungen

Mit den Netzwerkeinstellungen wurden Sie bereits in Abschnitt 1.7.1 konfrontiert, falls Sie CDT mit Hilfe der Funktion *Software Updates...*

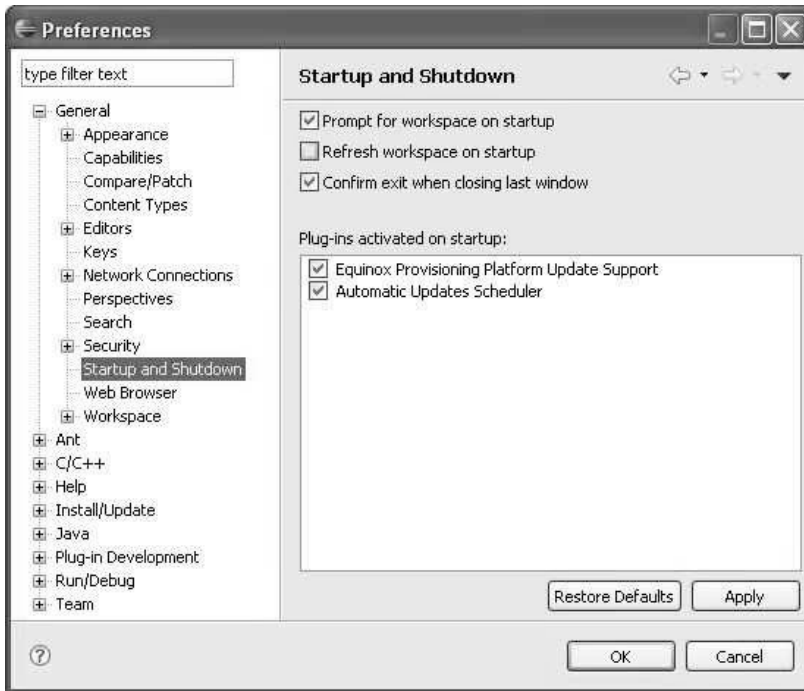


Abb. 2-19
Voreinstellungen, die
das Starten und
Beenden von Eclipse
betreffen

installiert haben. Zusätzlich zu der Aktualisierungsfunktion von Eclipse greifen auch andere Plugins auf die Einstellungen zurück, wie z.B. die Subversion-Unterstützung. Hauptsächlich bedürfen sie einer Änderung, wenn Sie keine direkte Internet-Verbindung haben, sondern nur über einen Proxy auf die Ressourcen des Internets zugreifen können. In Abbildung 2-20 haben Sie die Ansicht der Voreinstellung noch einmal dargestellt.

Netzwerkeinstellungen werden Workspace-übergreifend im Verzeichnis `configuration` abgelegt. Das bedeutet, wenn Sie einen neuen Workspace anlegen, so müssen Sie die Einstellungen nicht erneut vornehmen, was auch ziemlich sinnvoll ist.

2.5.6 Einstellungen für SSH-Client

Das Eclipse-Framework besitzt einen integrierten SSH2-Client, der z.B. von der CVS-Unterstützung, aber auch von anderen externen Plugins verwendet werden kann. Einstellungen für diesen nehmen Sie auf der Seite *General > Network Connections > SSH2* vor. Auf der Seite befinden sich drei Register, von denen Sie zwei in Abbildung 2-21 vorfinden.

Im ersten Register gibt's ganze zwei allgemeine Einstellungen vorzunehmen. Zum einen das Verzeichnis der SSH2-Schlüssel (*SSH2-Home*),

Abb. 2-20
Netzwerkeinstellungen

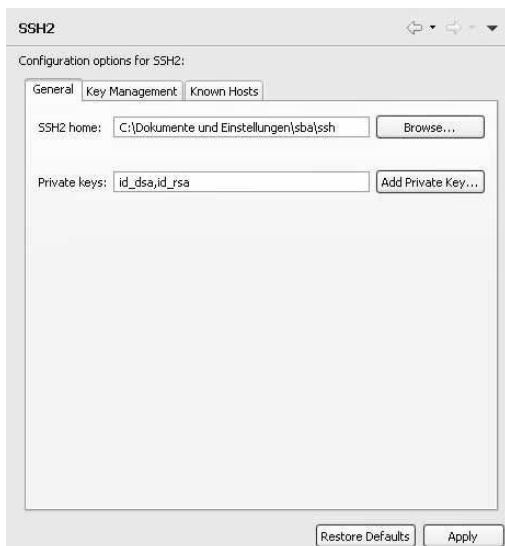
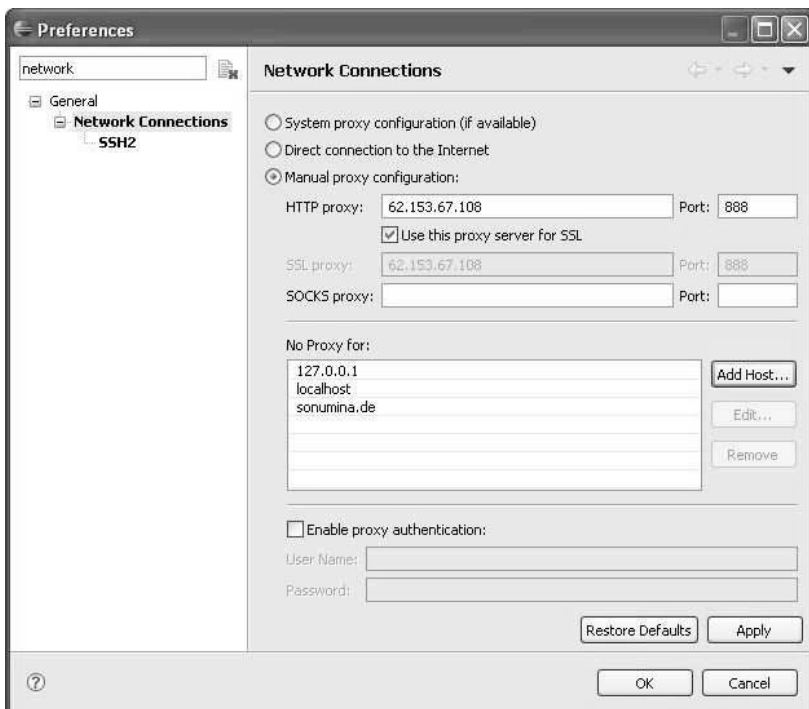


Abb. 2-21
SSH2-Einstellungen

zum anderen die privaten Schlüssel (*Private keys*), die bei der Initialisierung der Verbindung benutzt werden sollen.³

Eine komfortable Möglichkeit, ein neues Schlüsselpaar zu erzeugen, wird gleich im Register *Key Management* geboten. Mit den oberen Buttons können Sie ein neues DSA- oder ein neues RSA-Schlüsselpaar generieren lassen sowie bereits vorhandene Schlüssel laden. Gleich unter den Buttons befindet sich ein Textfeld, in dem sich der öffentliche Teil des aktuellen Schlüsselpaars befindet. Den Inhalt gilt es der Datei *authorized_keys* des Zielrechners hinzuzufügen, die sich dort für gewöhnlich ausgehend vom Heimatverzeichnis im Verzeichnis *.ssh* befindet. Ganz einfach geht das per Klick auf den Button *Export Via SFTP...*, woraufhin nach Angabe des Zielrechners und Login der Schlüssel in die Datei ohne weiteres Zutun eingefügt wird. Den privaten Schlüssel können Sie schließlich mit dem Button *Save Private Key...* auf Ihren lokalen Rechner bannen. Auf Wunsch können Sie den Zugriff auf den Inhalt zusätzlich durch eine Passphrase sichern, die Sie dann unter *Passphrase* und *Confirm passphrase* einzugeben haben.

2.5.7 Passwortspeicher

Nicht immer möchte man das Passwort neu eingeben. Deswegen bietet Eclipse einen Passwortspeicher an. Damit die Sicherheit nicht allzu sehr in Mitleidenschaft gezogen wird, wird ein Master-Passwort benutzt, um den Passwortspeicher zu verschlüsseln. Dessen Eigenschaften können Sie auf der Voreinstellungsseite *General > Security > Security Storage* modifizieren.

2.5.8 Komponenten (de-)aktivieren

Wenn Sie in Version 3.4 von Eclipse das Set der installierten Features modifizieren, so hat das Auswirkungen auf alle in diesem Kontext gestarteten Instanzen von Eclipse, unabhängig davon, auf welchem Workspace diese operieren. Dies ist nicht immer erwünscht. Befinden sich in einem Workspace ausschließlich C/C++-Projekte, so benötigen Sie das JDT-Plugin für Java-Projekte dort wahrscheinlich nicht. Sofern dies von den Plugins unterstützt wird, können Sie einzelne Komponenten aber im Kontext eines Workspace abschalten.

Hierfür bietet die Seite *General > Capabilities* in den Voreinstellungen diverse Möglichkeiten, deren Ansicht in Abbildung 2-22 dargestellt wird. Präsentiert wird die Liste *Capabilities*, die alle Kategorien von Komponenten auflistet. Mit der vorangestellten Auswahlbox legen Sie fest, ob diese Komponente im Workspace aktiviert sein soll oder nicht.

³Dem Gegenpart müssen entsprechende öffentliche Schlüssel bekannt sein.

Entfernen Sie beispielsweise das Häkchen vor *Team*, so wird Eclipse nicht mehr in der Lage sein, mit CVS oder ähnlich eingeordneten Versionierungssystemen umzugehen.

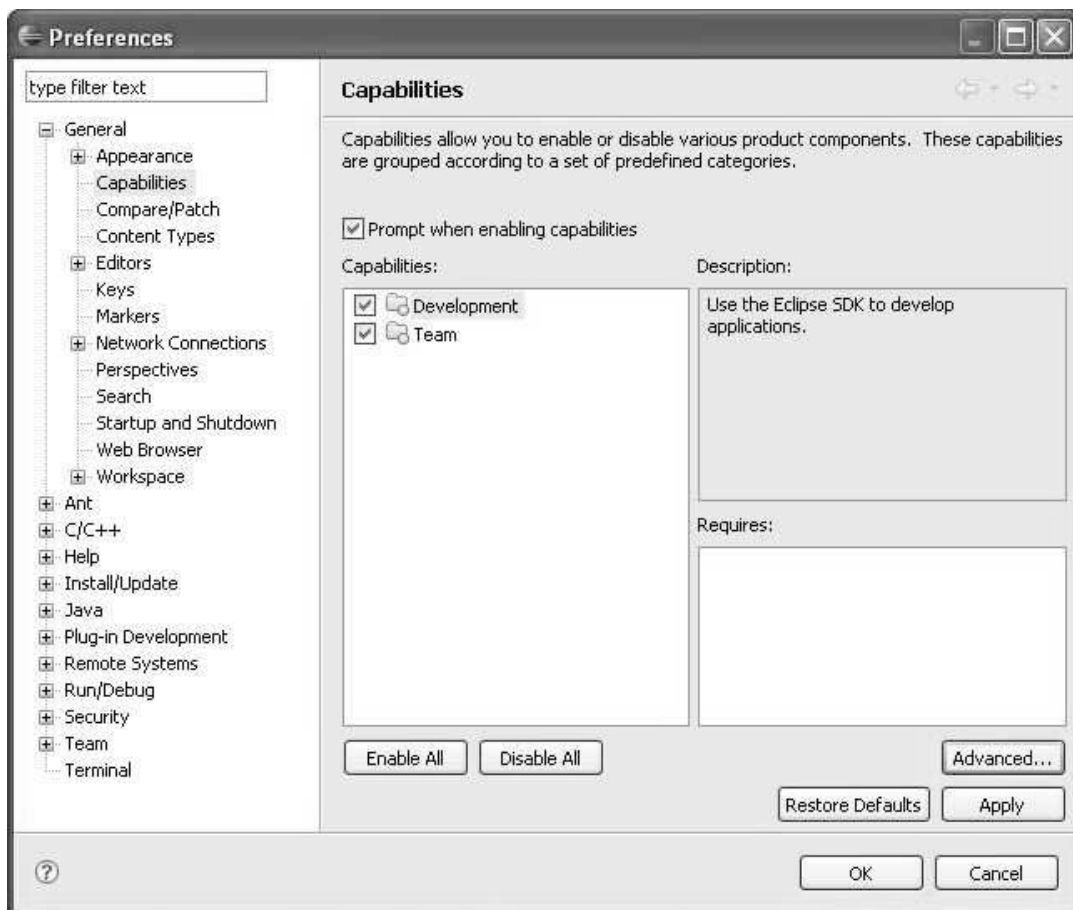


Abb. 2-22
Einstellungen der
aktiven Komponenten

Eine feinkörnigere Auswahl über die Komponenten erhalten Sie, wenn Sie den Button *Advanced...* anklicken. Dann werden die bisher angezeigten Komponenten weiter unterteilt. Sie könnten damit z.B. die CVS-Unterstützung ausschalten, während andere Versionierungssysteme noch aktiv blieben. Auch könnten Sie die Java-Unterstützung explizit deaktivieren, da auch JDT dieses Ausschalt-Feature unterstützt. Leider finden derzeit nur wenige andere Komponenten ihren Weg in diese Voreinstellung. So wird auch CDT noch nicht dort aufgelistet. Das bedeutet, dass dessen Aktivität auf diese Weise derzeit noch nicht beeinflusst werden kann.

2.6 Die Benutzerschnittstelle anpassen

Zum Abschluss des Kapitels und des ersten Teils des Buchs erfahren Sie noch, welche Möglichkeiten bestehen, die Oberfläche an Ihre eigenen Wünsche anzupassen. Hier werden bedienungsrelevante und auch ein wenig kosmetische Einstellungen besprochen, mit denen Sie beispielsweise auch das Layout der Workbench festlegen können. Falls Sie mit der Ansicht, die Eclipse Ihnen bietet, im Großen und Ganzen zufrieden sind, können Sie diesen Abschnitt auch überspringen.

2.6.1 Werkzeug- und Menüleiste anpassen

Die Werkzeugleiste wird immer im oberen Teil unter der Menüleiste dargestellt. Von dieser Restriktion einmal abgesehen, können Sie die Symbole der Leiste nach Belieben anordnen, auch auf mehrere Zeilen umspannende Weise. Oder Sie lassen sie mit dem Kontextmenüeintrag *Hide Toolbar* gänzlich verschwinden. Wenn das einmal passiert ist, können Sie dies mit dem Menüeintrag *Window > Show Toolbar* auch wieder rückgängig machen.

Aktionen, die über Menü- und Werkzeugleiste aufrufbar sind, hängen von der dargestellten Perspektive ab. Im Allgemeinen ist die Auswahl durch den Anbieter des zur Perspektive zugehörigen Plugins sinnvoll vorgegeben. Sie kann von Ihnen aber auch modifiziert werden. Hierfür wählen Sie den Menüeintrag *Window > Customize Perspective...*, woraufhin sich ein neues Fenster öffnet, das aus zwei Registern besteht, die im Folgenden beschrieben werden.

Shortcut

Das Register *Shortcut* hat das in Abbildung 2-23 gezeigte Aussehen. Hier können Sie bestimmen, welche Einträge in den Shortcuts der Workbench erscheinen. Derzeit existieren drei Menüpunkte, die Shortcuts aufnehmen können. Das wären:

- *New*. Das Menü *Project > New*, um eine neue Ressource anzulegen.
- *Open Perspective*. Das Menü *Window > Open Perspective*, um eine Perspektive zu öffnen, das auch über den Button links neben den bereits geöffneten Perspektiven verfügbar ist.
- *Show View*. Das Menü *Window > Show View*, um einen View zu öffnen. Auch das Menü *Show View as fast view*, das sich standardmäßig in der Workbench links unten in der Statusleiste befindet, beinhaltet die ausgewählten Shortcuts.

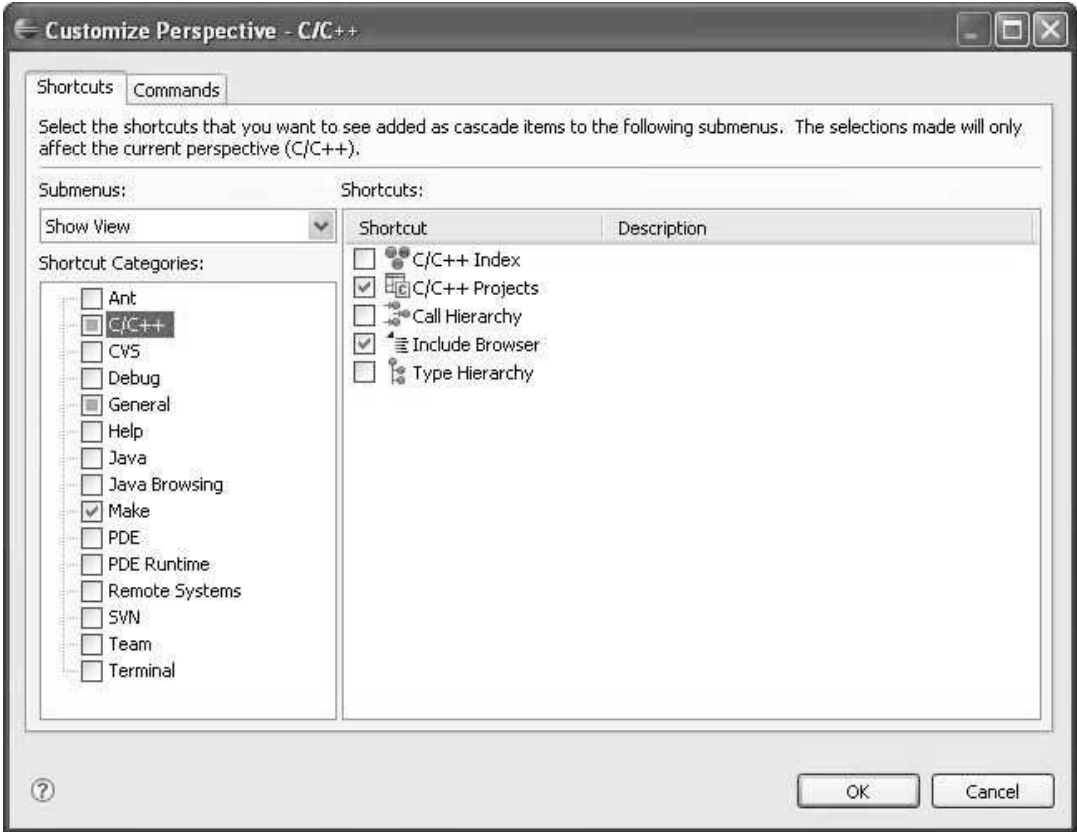


Abb. 2-23

Das Register Shortcut

Links im Register werden in einem Baum die unterschiedlichen Kategorien präsentiert, in die die Shortcuts thematisch eingeteilt werden können. Das Auswählen einer Kategorie lässt einzelne Elemente in der rechten Tabelle erscheinen. Aktivieren Sie dort die Elemente, die als Shortcuts im gewählten Untermenü fungieren sollen.

Commands

Das Register *Commands* ist in Abbildung 2-24 dargestellt. Hier wird festgelegt, welche *Kommandogruppen* in der Perspektive verfügbar sind. Eine Kommandogruppe umfasst dabei Aktionen, die zu der gleichen Kategorie passen, und kann Aktionen für das Hauptmenü (mittlere Tabelle) oder Aktionen für die Werkzeugleiste (rechte Tabelle) beinhalten.

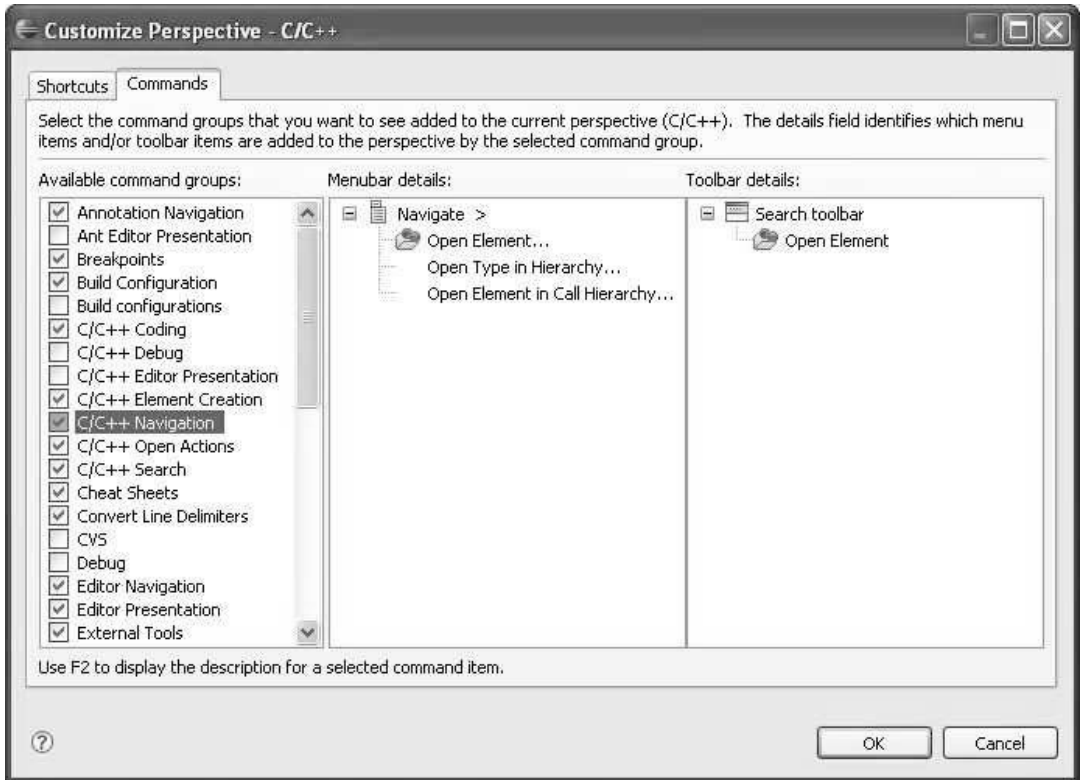


Abb. 2-24
Das Register
Commands

In der linken Tabelle werden alle verfügbaren Gruppen dargestellt. Selektieren Sie ein Element, so zeigen die anderen beiden Tabellen an, wie sich die Kommandogruppe manifestiert. Wenn Sie diese Gruppe dann mit Hilfe des Häkchens aktivieren, dann werden die Kommandos der Perspektive tatsächlich hinzugefügt. So hätte eine aktivierte Kommandogruppe *C/C++ Navigation* zur Folge, dass die Einträge *Open Element...*, *Open Type in Hierarchy...* und *Open Element in Call Hierarchy...* im Workbench-Menü *Navigate* erscheinen und der Button *Open Element* in der Werkzeugleiste erscheint.

2.6.2 Perspektiven verwalten

Neben den bereits vorgefertigten Perspektiven, die durch Plugins dem Eclipse-Framework hinzugefügt werden, können Sie beliebig viele eigene Perspektiven mit eindeutigen Namen erstellen. Hierfür wählen Sie den Menüeintrag *Window > Save Perspective As...* aus, woraufhin ein

Fenster eine Auswahl aller vorhandenen Perspektiven präsentiert und zusätzlich ein Eingabefeld für einen Namen anbietet.

*Perspektiven anlegen
oder überschreiben*

Wählen Sie im Fenster einen neuen Namen, so wird eine neue Perspektive unter diesem Namen angelegt; ansonsten wird eine bereits existierende Perspektive überschrieben. Abgespeichert werden die üblichen Parameter einer Perspektive, d.h. welche Views geöffnet sind, welche Positionen sie in der Workbench einnehmen und über welche Kommandos verfügt werden kann. Das Symbol einer neuen Perspektive entspricht im Übrigen dem Symbol der Perspektive, aus der sie hervorgeht.

*Perspektiven entfernen
oder wiederherstellen*

Möchten Sie eine Perspektive löschen, so kommen Sie mit diesem Vorhaben über die Voreinstellungen auf der Seite *General > Perspectives* weiter. Klicken Sie die zu löschende Perspektive in der Liste an und anschließend auf *Delete*. Es können allerdings nur selbst erstellte Perspektiven gelöscht werden. Wünschen Sie dagegen den Ursprungszustand einer überschriebenen Perspektive wiederherzustellen, so benutzen Sie den Button *Reset*, wenn diese Perspektive angewählt ist.

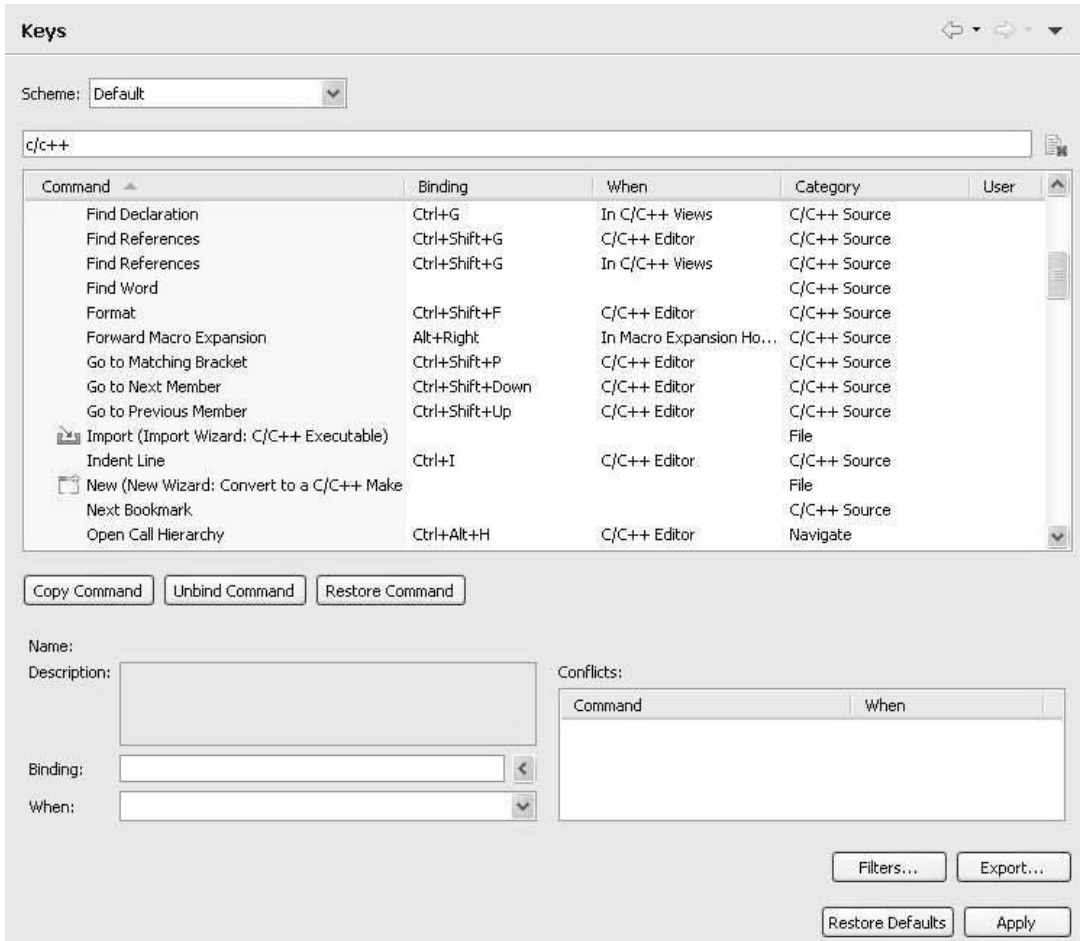
2.6.3 Tastatursteuerung anpassen

Eclipse lässt sich komplett über die Tastatur steuern. Dabei sind die Tastenkürzel frei von Ihnen definierbar. Hierzu bedienen Sie sich in den globalen Voreinstellungen der Seite *General > Keys*. Diese ist in Abbildung 2-25 dargestellt.

Den größten Teil der Seite beansprucht eine Liste, in der alle Kommandos (Spalte *Command*), ihre aktuelle Zuordnung auf der Tastatur (Spalte *Binding*), in welchem Kontext sie gültig sind (Spalte *When*) und ihre Kategorie (Spalte *Category*) aufgezeigt werden. Die letzte Spalte *User* gibt an, ob Sie an diesem Eintrag Änderungen durchgeführt haben (angezeigt durch ein *U*) und wenn ja, ob es sich dabei um einen Konflikt handelt (angezeigt durch ein *C*). Ein Konflikt tritt dann auf, wenn ein und dasselbe Tastenkürzel im gleichen Kontext zwei unterschiedlichen Kommandos zugewiesen worden ist.

Die Liste ist insgesamt sehr groß, bietet aber die Möglichkeit, Einträge nach den Spalten zu sortieren. Viel wirkungsvoller ist freilich der Filter, der mit Hilfe des über der Liste befindlichen Texteingabefeldes angegeben wird und nur diejenigen Elemente anzeigt, die irgendwo die eingegebene Zeichenkette enthalten.

Wenn Sie einen Eintrag in der Liste anklicken, dann können Sie Änderungen an diesem Kommando durchführen. Zunächst erscheint in *Description* eine kurze Beschreibung des Kommandos. In *Binding* spezifizieren Sie dann das Tastenkürzel, bei dessen Betätigung das selektierte Kommando ausgeführt werden soll. Dieses Feld ist kein gewöhn-

**Abb. 2-25**

Tastatursteuerung anpassen. Dargestellt ist die Voreinstellungsseite ohne Auswahl.

liches Texteingabefeld, sondern beantwortet Ihre Eingaben mit einer Beschreibung der gedrückten Taste.

Über den Filter sehen Sie noch eine Auswahlbox, die den Titel *Scheme* trägt. Darunter verbergen sich vordefinierte Zuweisungen von Tastaturkürzeln der Kommandos. Diese sind nach Eclipse-Art selbstverständlich nicht fest in ihrer Anzahl: Plugins können weitere Schemata hinzufügen. So haben Sie als C/C++-Entwickler mindestens die Auswahl zwischen

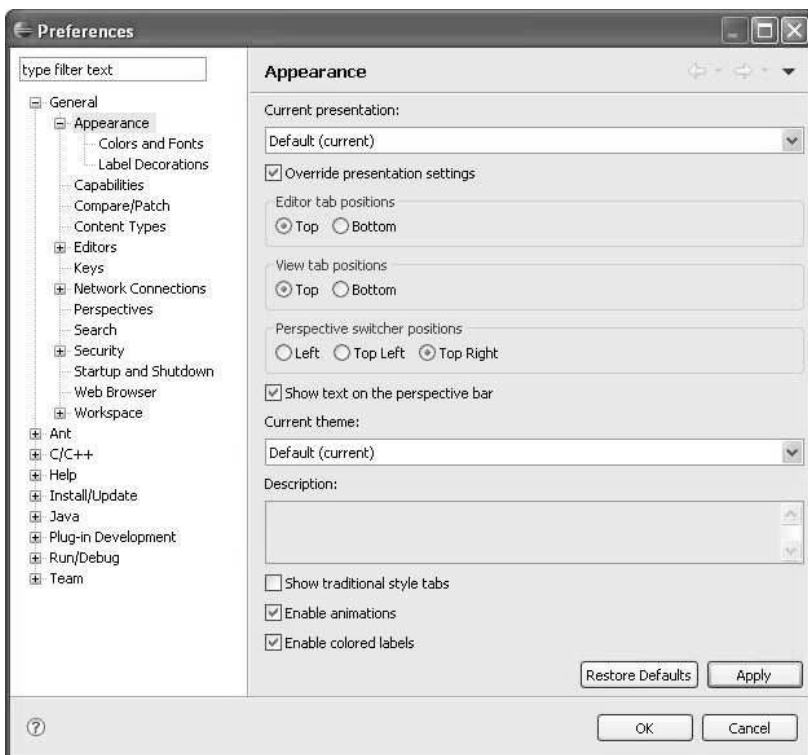
- *Default*,
- *Emacs* und
- *Microsoft Visual Studio*.

Letzteres Schema ist freilich vom C/C++-Plugin hinzugefügt worden und passt die Tastatursteuerung an eine *Microsoft Visual Studio* ähnliche Variante an.

2.6.4 Erscheinungsbild

Zum Abschluss des ersten Teils sollen noch ein paar Worte über die kosmetische Flexibilität von Eclipse verloren werden. Hier ist die Voreinstellungsseite *General > Appearance* der Ausgangspunkt. Die möglichen Optionen sehen Sie in Abbildung 2-26.

Abb. 2-26
Erscheinungsbild
ändern



Im oberen Bereich der Seite können Sie das *Look and Feel* der Workbench festlegen und damit letztendlich das Aussehen der Editoren und Views bis hin zu ihren grundsätzlichen Bedienungsweisen beeinflussen. Die Auswahlliste *Current presentation* hält hierfür Einträge aller verfügbaren Skins bereit. Da das Aussehen von Eclipse über die Jahre stetigen Veränderungen unterworfen war, werden standardmäßig neben der bereits bekannten Präsentation lediglich solche Skins angeboten, die Eclipse ein Aussehen aus früheren Zeiten verleihen.

Wenn Sie die Option *Override presentation settings* aktivieren, können Sie einige Parameter des ausgewählten Skins überschreiben. Diese betreffen die Positionen der Tabs und der Knöpfe für geöffnete Perspektiven.

Im unteren Bereich können Sie unter *Current theme* ein Themenset auswählen und damit Farben und Zeichensätze für verschiedene Arten von Ausgaben bestimmen. Im Detail können Sie diese in der untergeordneten Einstellungsseite *Color and Fonts* anpassen.

- *Show traditional style tabs*. Lässt Sie das Erscheinungsbild der Tabs auf eine weniger geschwungene Form einstellen.
- *Enable animations*. Das Öffnen und Verstecken von Views wird animiert.
- *Enable colored labels*. Wenn diese Option aktiviert ist, dann erscheinen Einträge in Listen und Bäumen farbig, um bestimmte Informationen besonders hervorzuheben. Auf welche Weise dies geschieht, können Sie ebenfalls in *Color and Fonts* einstellen. Gebrauch von farbigen Einträgen machen der *Search-View*, Text-Dekorationen und viele andere Elemente.

In der Tat können sich auch in dieses System Plugins einhängen und weitere Präsentationen hinzufügen. Als Beispiel sei hier das *Extended VS Presentation*-Plugin erwähnt, das versucht, das Feeling von Visual C++ auf Eclipse zu übertragen⁴. Dessen Heimatseite ist unter der Adresse

Skin-Plugins

<http://andrei.gmxhome.de/skins/index.html>

zu finden. Die Update-Site lautet:

<http://andrei.gmxhome.de/eclipse/>.

Nachdem die Installation des Plugin erfolgreich abgeschlossen ist, erscheint in der Auswahlbox unter *Current presentation* der neue Eintrag *Extended VS Presentation*. Einen Eindruck von dem Aussehen können Sie sich anhand Abbildung 2-28 machen.

2.7 Verzeichnislayout auf dem Datenträger

Dieser Abschnitt möchte Ihnen die Inhalte und Organisationsstruktur der Verzeichnisse näherbringen, die bei der Entwicklung unter Eclipse eine Rolle spielen. Idealerweise sollten Ihnen – als Anwender von Eclipse – die meisten der hier vorgestellten Details verborgen bleiben

⁴Ob so etwas wirklich sein muss, ist freilich Geschmackssache.

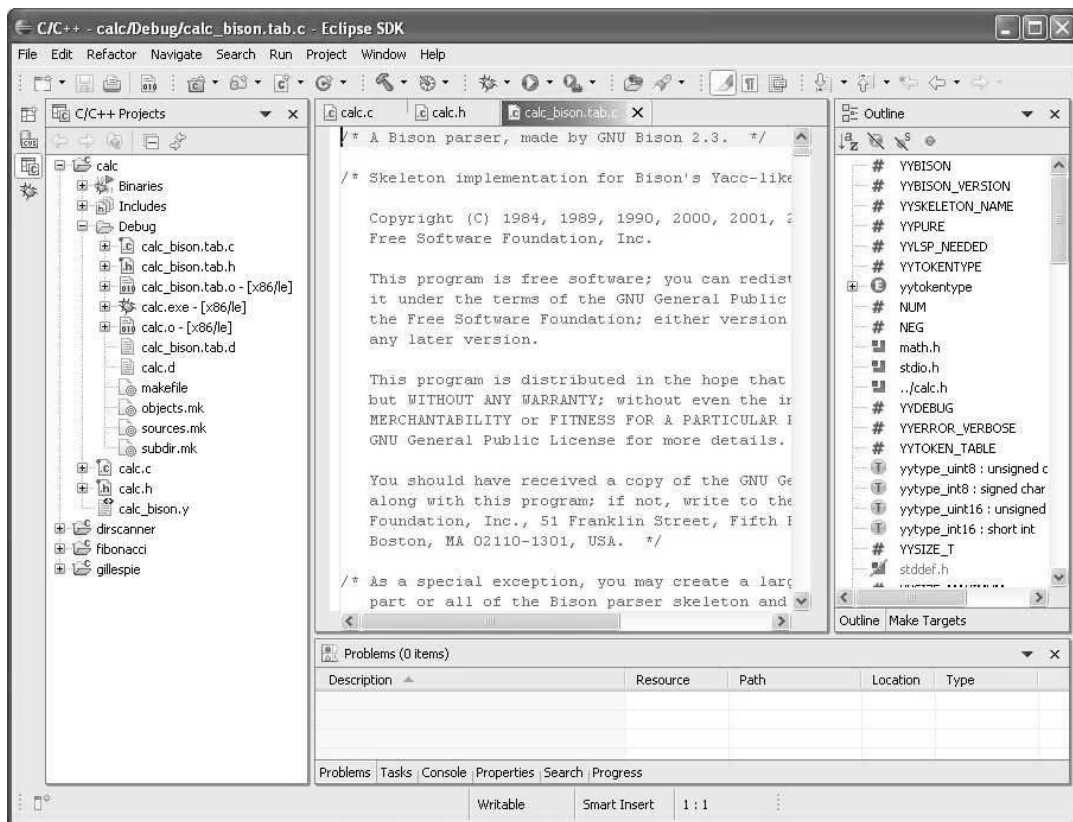


Abb. 2-27
Eclipse im alten
Gewand

und deshalb zunächst kaum von Interesse sein. In der Realität kommen Sie jedoch nicht um ein tieferes Verständnis herum, denn Eclipse selbst ist eine äußerst komplexe Software, die, obwohl ausgiebig getestet, Ihnen ab und zu Probleme bereiten wird, zum Beispiel in Form von Abstürzen. Jegliches Hintergrundwissen über die systemnahe Struktur kann Ihnen dabei helfen, zumindest die Ursachen für ein Problem zu finden oder in zweiter Instanz dieses zu beheben.

Wichtige Verzeichnisse sind dabei das Programmverzeichnis sowie die Verzeichnisse der Arbeitsbereiche. Im Folgenden wird auf deren Bedeutung und deren Handhabung eingegangen.

2.7.1 Programmverzeichnis

Das Programmverzeichnis ist dasjenige, das Sie nach dem Entpacken des heruntergeladenen Eclipse-Archivs vorliegen haben. Sie wissen be-

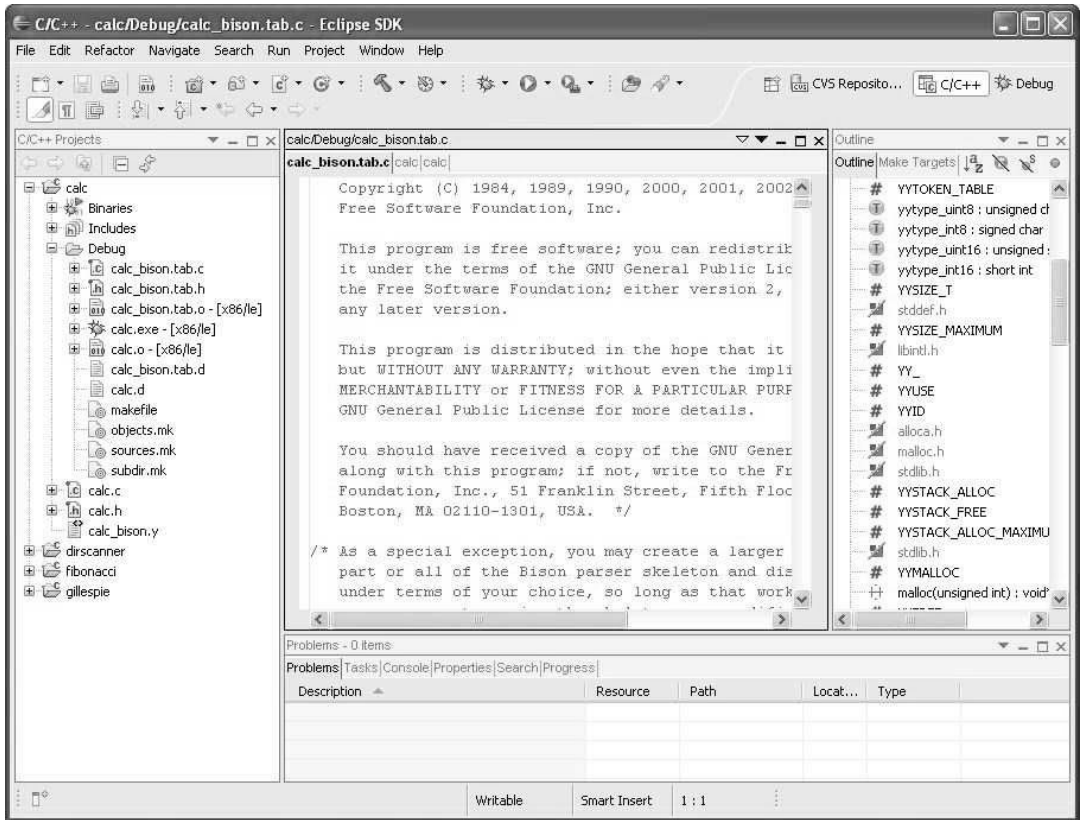


Abb. 2-28

Eclipse mit dem Visual Studio Theme

reits, dass das Verzeichnis je nach Betriebssystem die Dateien eclipse oder eclipse.exe beinhaltet. Am Anfang des Buchs wurden sie als Eclipse Program Launcher bezeichnet. Sie wissen auch bereits über die Datei eclipse.ini Bescheid, mit deren Hilfe Sie den Start von Eclipse beeinflussen können. Zusätzlich verbergen sich freilich noch weitere Dateien oder Verzeichnisse darin, die für Sie in Abbildung 2-29, einem Screenshot vom Windows Explorer, aufgelistet werden.

Verzeichnis: configuration

Hier werden die in Eclipse vorgenommenen Einstellungen in Form von Konfigurationsdateien abgelegt. Bequemerweise werden die Dateien im ASCII-Format ausgeschrieben, so dass sie in jedem beliebigen Editor angezeigt oder gegebenenfalls modifiziert werden können. Allerdings

**Abb. 2-29**

Die Ansicht des Programmverzeichnis nach dem Entpacken

landen in diesem Verzeichnis nur Workspace-übergreifende Einstellungen, wie z.B. die der Netzwerkeinstellungen.

Von besonderer Bedeutung ist die Datei `bundles.info`, die im Unterverzeichnis `org.eclipse.equinox.simpleconfigurator` lokalisiert ist. Darin werden alle während des Programmstarts zu aktivierenden Plugins aufgezählt. Sie sollten an ihr nur im Notfall Hand anlegen, da Sie von Eclipse – oder besser von `p2` – verwaltet wird.

Verzeichnis: dropins

Dieses Verzeichnis gehört zu den überwachten Verzeichnissen (engl. *watched directory*). Überwachte Verzeichnisse werden beim Start von Eclipse eingelesen, um nach noch nicht eingerichteten Features und Plugins Ausschau zu halten, die dann dem System hinzugefügt werden.

Verzeichnis: p2

Das Verzeichnis enthält die sogenannte Profil-Registry, in der alle aktuellen Installationen aufgelistet sind.

Verzeichnis: plugins

In diesem Verzeichnis befinden sich sämtliche Plugins. Nahm dieses Verzeichnis vor Version 3.4 auch manuell hinzugefügte Plugins auf, so ist dies jetzt eigentlich tabu. Das Konzept des *dropins*-Verzeichnis ist das weitaus flexiblere, da es eine saubere Trennung von benutzerhinzugefügten Plugins und standardmäßig vorhandenen Plugins erlaubt.

Verzeichnis: features

Ein *Feature* fasst Plugins, die vom selben Anbieter sind und sich zu einem Werkzeug ergänzen, zu einer in Eclipse verwaltbaren Einheit zusammen. Verwaltbar bedeutet, dass diese installiert oder deinstalliert werden können. Features können auch an Voraussetzungen gebunden sein, z.B., dass die Präsenz eines anderen Features gefordert wird. Die Infrastruktur in Form von Ressourcen landet in diesem Verzeichnis. Wie beim *plugins*-Verzeichnis ist es ab Eclipse 3.4 tabu, manuell Hand anzulegen.

Verzeichnis: readme

Hier finden Sie Copyright-Informationen oder Ähnliches von Eclipse und weiteren Plugins.

Sonstige Dateien

Neben den bereits vorher erwähnten Dateien `eclipse.ini` und dem Eclipse Program Launcher finden Sie auf einer Windows-Installation die Datei `eclipse.exe` im Programmverzeichnis vor. Diese sollten Sie anstelle von `eclipse.exe` benutzen, wenn Sie Eclipse über die Windows-Kommandozeile zu starten wünschen. Damit funktionieren dann auch Dateiumleitungen, wenn Eclipse in die Standardausgabe schreibt.

2.7.2 Workspace

Das Workspace-Verzeichnis ist das Verzeichnis, in dem Eclipse alle benutzerrelevanten Daten ablegt. Zum einen sind das natürlich die Projekte, deren von Ihnen angelegte Struktur dort eins zu eins abgebildet wird. Zum anderen existiert ein Verzeichnis mit Namen `.metadata`.

- log Die Datei `.metadata/log` beispielsweise enthält alle geloggte Ausgaben vom Eclipse-Framework. Sollte es schwerwiegende Probleme wie Abstürze in Eclipse geben, so ist das der wohl am besten geeignete Ort, um mit der Ursachenforschung zu beginnen.