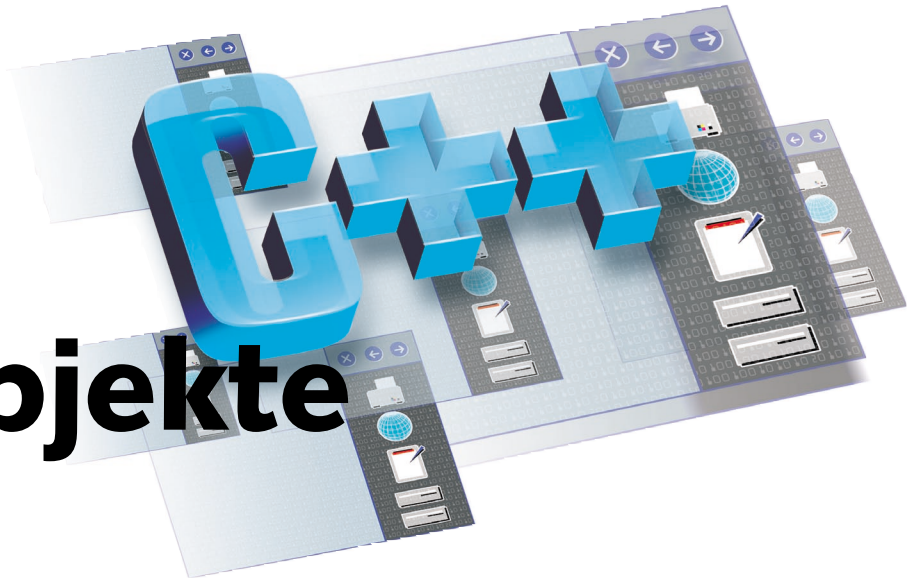




Das Common Object Model

Es lebt: COM-Objekte mit ATL



Mit **COM** programmieren Sie Objekte, die auch in anderen Programmen benutzt werden können. Das ist **ungeheuer praktisch**, allerdings ist COM nicht **ganz leicht** zu beherrschen.

THOMAS WÖLFER

COM ist cool, das sollte zu Anfang mal gesagt werden. COM-(Common Object Model) Objekte können in praktisch alle kompilierten Programmiersprachen geschrieben werden, und C++ ist da keine Ausnahme. Der Grund dafür ist der, dass es sich bei COM mehr um eine Spezifikation und weniger um eine Technologie handelt. In diesem Beitrag erfahren Sie, wie Sie eigene COM-Objekte bauen, ohne sich um die Komplexität von COM selbst kümmern zu müssen.

Die Spezifikation beschreibt dabei das Speicherlayout, das ein Objekt haben muss, um als COM-Objekt zu gelten, sowie einen Satz Funktionen, die die COM-Infrastruktur bilden. Ferner legt COM noch die Art und Weise fest, mit der mit einem COM-Objekt kommuniziert werden kann und in der das Objekt auf diese Kommunikationsversuche reagieren soll. Das hört sich ein bisschen kompliziert an, und das ist es auch, daher sollen einige der wichtigsten Grundlagen von COM zunächst erläutert werden.

■ COM-Objekte im Speicher

Die Infrastruktur für COM wird von den COM-Funktionen aus der COM-Library zur Verfügung gestellt. Diese Infrastruktur stellt verschiedene Dinge

bereit, die für COM notwendig sind. Um sie zu verstehen, muss man sich vergegenwärtigen, wie ein Programm ein COM-Objekt verwendet.

Zunächst einmal läuft irgendwo ein Programm, das gerne der Client eines COM-Objekts sein möchte. Dazu muss dieses Programm in der Lage sein, das Objekt auf dem lokalen Rechner (oder im Netz) zu finden, denn das Objekt liegt irgendwo in binärer Form als Datei auf der Festplatte vor. Dazu kann das Programm zum Beispiel die COM-Infrastruktur verwenden, denn COM stellt einen eigenen Mechanismus zum Auffinden von Objekten zur Verfügung.

Dazu sind alle COM-Objekte mit einem jeweils eindeutigen Identifizierer versehen. Bei der Windows Implementierung von COM ist dieser Identifizierer einfach in der Registry eingetragen und verweist dort auf die ausführbare Datei, die das Objekt enthält.

Dann muss der COM-Client das Objekt laden. Dabei stellt sich die Frage, in welchen Adressraum das Objekt geladen wird: in einen eigenen, in den des aufrufenden Programms, oder in einen von mehreren Programmen gemeinsam genutzten, das sind dabei die Optionen. Dabei wird es bei verteiltem COM übers Netz noch etwas schwieriger, denn hier kann ein COM-Objekt sogar auf einem anderen Rechner laufen. Es befindet sich also noch nicht einmal im physisch gleichen RAM, den das COM-Client-Pro-

gramm nutzt. Auch diese Problematik wird von der COM-Infrastruktur gelöst: Diese stellt im Wesentlichen einen Proxy-Dienst zur Verfügung, über den mit dem tatsächlichen Objekt kommuniziert werden kann. Das Objekt selbst wird dann vom Client nur über einen speziellen Eintrittspunkt, der "Interface" genannt wird, angesprochen.

Ist das Objekt geladen, will der COM-Client natürlich die Dienste des Objekts benutzen. Dazu exportiert das Interface des COM-Objekts Methoden. Damit diese aber irgendetwas Sinnvolles tun können, müssen diese Methoden zumindest Parameter erhalten können, und das sowohl per Referenz als auch per Value. Auch das ist nicht ganz einfach, denn hier sind die unterschiedlichsten Dinge zu beachten.

Bei Zeiger-Parametern ist es zum Beispiel so, dass das COM-Objekt in der Theorie in einen Speicherbereich schreiben müsste, der sich – um das COM im Netzbeispiel beizubehalten – auf einem ganz anderen Rechner befindet, der unter anderem auch noch eine andere Systemarchitektur und damit ein anderes Byte-Ordering hat.

Auch dazu stellt die COM-Library die Infrastruktur bereit: Der Client ruft einfach nur die gewünschte Methode auf, aber die COM-Library bekommt dann viel Arbeit. Sie alloziert zunächst genug Speicher in einem zum COM-Objekt lokalen Bereich, und sorgt dann



dafür, dass der Speicher auch für das Objekt im richtigen Layout vorliegt. Schließlich ruft es erst mit diesem fertig verpackten Block die echte Methode im COM-Objekt auf. Wenn diese terminiert ist, wird der vorherige Vorgang wieder in umgekehrter Reihenfolge durchgeführt und schließlich erhält der COM-Client das gewünschte Ergebnis.

Sowohl für den Client als auch für das Objekt sind diese Transaktionen völlig transparent – zumindest fast, denn es kommen noch ein paar Schwierigkeiten hinzu, die nicht von der COM-Library abgehandelt werden und teilweise auch nicht abgehandelt werden können. Das sind im Wesentlichen das Reference-Counting und der Support für bestimmte Sprachen. Nachdem eingangs erwähnt wurde, dass man COM-Objekte in praktisch jeder kompilierten Sprache erstellen kann, mag es nun ein wenig verwundern, dass der "Support" für bestimmte Sprachen ein Problem darstellt.

Das ist aber einfach erklärt. Angenommen, Sie haben ein COM-Objekt, das eine Methode zum Umrechnen zwischen zwei Währungen anbietet, zum Beispiel zwischen EUR und DM, wird

Das ist wunderbar und praktisch. Was passiert jedoch, wenn Sie möchten, dass Ihr COM-Objekt auch von Clients benutzt werden soll, die nicht in C++ geschrieben wurden? Schließlich gibt es Programmiersprachen, die gar keine Zeiger haben – es gibt sogar welche, die keinen Datentyp für "double" besitzen! Nun könnte man sich natürlich auf den Standpunkt stellen, dass man derartig absurde Sprachen einfach nicht unterstützt – nur sind diese Sprachen gar nicht so absurd, sondern im Gegenteil extrem weit verbreitet. Auf jeder Windows-Installation findet sich zum Beispiel VBScript, und VBScript hat beispielsweise keine Zeiger – und schon gar keine explizit spezifizierten Datentypen. VBScript ist aber trotzdem sehr praktisch und sollte von einem COM-Objekt nicht von vornherein ausgeschlossen werden. Das gilt im Übrigen auch für VBScript und JavaScript, das auf Webseiten verwendet werden kann, oder für die verschiedenen Sprachen, die in ASP-Seiten verwendet werden können. Sollen diese tatsächlich alle ausgeschlossen werden? – In diesem Fall würde letzten Endes nicht viel mehr übrig bleiben, als C und C++ – und das wäre dann vielleicht doch ein bisschen wenig.

einschließlich der Verwendung des Speichers, berücksichtig. Mit anderen Worten: Man stellt beim COM-Objekt keine Methode *EuroToDM(double eur, double* dm)* zur Verfügung, sondern statt dessen eine Methode *EuroToDM(VARIANT eur, VARIANT dm)*.

Der einzige Trick ist dann noch herauszufinden, in welchem Member der eingehenden Variant-Struktur die gewünschten Daten tatsächlich liegen, und in welches Member der ausgehenden Struktur die Ergebnisse geschrieben werden müssen.

■ Reference Counting

Das Reference Counting ist ein weiteres Problem, das angegangen werden muss – allerdings nur im COM-Client: Reagiert werden muss allerdings im Objekt. Zunächst einmal stellt sich die Frage, was "Reference Counting" eigentlich ist. Dazu muss man sich das folgende Szenario vorstellen: Der COM-Client hat ein COM-Objekt gefunden und ein Interface dazu ermittelt. Dann hat der Client eine Methode des Interfaces verwendet. Wann kann aber nun das Objekt wieder aus dem Speicher entfernt werden? – Schließlich weiß das Objekt nichts davon, dass es gerade von einem bestimmten Client verwendet wird, bzw., dass ein bestimmter Client momentan einen Interface-Pointer gespeichert hat.

Dieses Problem wird mit dem Reference Counting gelöst. Dabei wird einfach jedes Mal ein Zähler hochgezählt, wenn ein Objekt referenziert wird. Wird die Referenz wieder aufgelöst (der Interface-Pointer entfernt), wird der Zähler um eins heruntergezählt. Ist der Zähler bei 0 angekommen, gibt es keine Referenz mehr, und das Objekt kann aus dem Speicher entfernt werden.

■ ATL kommt ins Spiel

Wie gesagt, kann man COM-Objekte mit praktisch allen Programmiersprachen erzeugen, auch mit C++. Bei VC++ können Sie COM-Objekte entweder direkt mit der COM-API in C programmieren, oder Sie programmieren eine eigene Lösung in C++, die Teile der Komplexität von COM kapselt. Dann gibt es auch noch die Möglichkeit, ein COM-Objekt mit den MFC zu programmieren und schließlich unterstützt auch die so genannte ATL (Active Template Library) die Erzeugung von COM-Objekten.

Dabei ist letzteres die beste Wahl, denn die ATL wurde speziell für die Pro-

LISTING 1

```
void EuroToDM( double euro, double * dm)
{
    *dm = euro / 2.0; // nicht ganz korrekter kurs...
}
```

das COM-Objekt in C++ programmiert. Diese Methode könnte wie in Listing 1 aussehen.

Wenn Sie nun ein COM-Client-Programm haben, das ebenfalls in C++ geschrieben ist, könnte dieses Programm

Allerdings bietet COM für dieses Dilemma ebenfalls eine Lösung an, und die kommt in Form des COM-Datentypen VARIANT. In Sprachen wie VBScript werden die einzelnen Parameter beim Aufruf einer Methode eines COM-Ob-

LISTING 2

```
void fooClient( IComObject* pI)
{
    double eur = 12.0;
    double dm;

    pI->EuroToDm( eur, &dm);
    printf( "%lf Euro sind %lf DM", eur, dm);
}
```

(angenommen, das Interface ist schon ermittelt) die Methode ganz einfach aufrufen (siehe Listing 2).

jekts von der COM-Library grundsätzlich in VARIANTs verpackt – und dabei ist die notwendige Konvertierung,

grammierung von COM-Objekten entworfen und wird genau für diesen Zweck auch mit Assistenten in der Visual C++-IDE unterstützt. Das Problem bei der ATL ist, dass diese Library auch nicht weniger komplex ist, als COM selbst. Das können Sie leicht überprüfen, da der Großteil des ATL-Quellcodes in Form von Header-Dateien vorliegt.

Ein kurzer Blick in die ATL-Header erschreckt einen meist: Die strotzen nur so vor Macros, Defines, und C++-Template-Klassen. Wirklich gut zu lesender Quellcode ist in der ATL bei weitem nicht zu finden – sie kann eher als eine Art Abschreckung für Einsteiger dienen.

Das ist aber kein Problem, denn man kann das, was die ATL tut, in allen wesentlichen Punkten zunächst völlig ignorieren. Stattdessen benutzt man die Library mit Hilfe der Assistenten und kümmert sich nicht weiter um den so erzeugten Quellcode. Stattdessen implementiert man nur die Funktionalität, die man haben will.

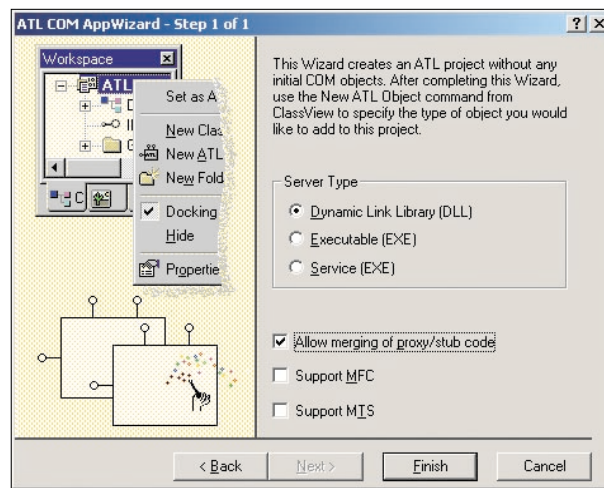
Irgendwann geht natürlich trotzdem kein Weg daran vorbei, sich mit den Spezifika der ATL auseinanderzusetzen – aber um ein einfaches COM-Objekt mit wenigen Methoden zu bauen, ist das

Schritt, allerdings sind die hier angebotenen Optionen doch erklärungsbedürftig.

Zunächst einmal können Sie einen "Server Type" auswählen. Dabei geht es darum, auf welche Art das COM-Ob-

nicht – für das Beispiel brauchen Sie das nicht. Unter dem "MTS" ist der "Microsoft Transaction Server" zu verstehen. Der bietet einige für COM-Objekte sehr interessante Dienste an, im Besonderen für COM-Objekte, die auf entfernten Rechnern im Netz liegen. Fürs Beispiel brauchen Sie diese Option aber nicht.

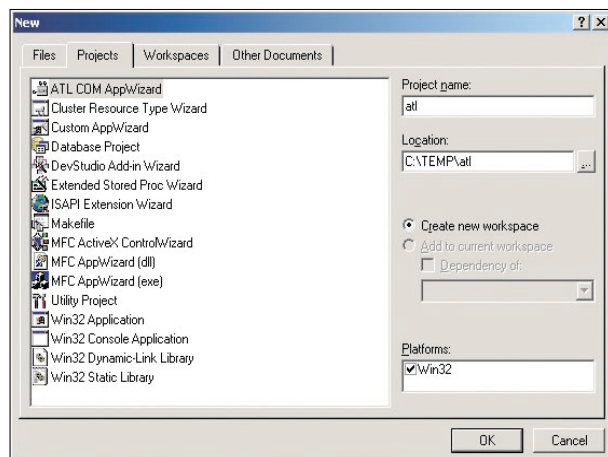
Der Wizard ist damit abgeschlossen, und es wird eine ganze Menge Quellcode erzeugt. Den ignorieren Sie allerdings. Zu diesem Zeitpunkt haben Sie schon die benötigte Infrastruktur für ein COM-Objekt – nur das Objekt selbst haben Sie noch nicht. Das bedeutet, Sie müssen nun ein neues



FÜRS BEISPIELPROGRAMM wählen Sie die Option "DLL" aus.

jekt transportiert werden soll – also ob es in einer .EXE- oder einer .DLL-Datei residieren soll – und um die Art des Objekts, denn Dienste (Services) als COM-Objekt sind nochmals anders zu behandeln.

es ATL-COM-Objekt in Ihr Projekt einfügen. Das geht mit der ClassView (Klassenansicht) aus dem Arbeitsbereich, die für ATL-Projekte eine deutlich größere Wichtigkeit hat, als das bei



EIN ATL-PROJEKT wird angelegt.

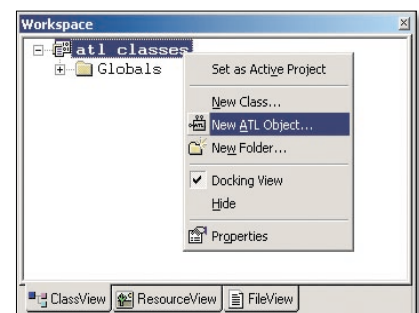
nicht notwendig. Stattdessen muss man aber wissen, wie die Assistenten zu bedienen sind, denn die sind nicht ganz so luxuriös, wie das bei den MFC-Gegenstücken der Fall ist.

Zunächst müssen Sie ein neues Projekt anlegen. Diesmal ist das Projekt vom Typ "ATL COM AppWizard", denn Sie werden ein COM-Projekt mit Hilfe von ATL realisieren. Der zugehörige Wizard hat nur einen einzigen

Dann können Sie noch auswählen, ob "proxy/stub" code "gemerged" werden soll, ob MFC unterstützt und ob MTS unterstützt werden soll. Dabei geht es um Folgendes: Sie wissen bereits, dass die Daten, die von der COM-Library zum Objekt transportiert werden, von der COM-Library umverpackt werden müssen, und Sie wissen auch, dass der COM-Client unter

Umständen nur mit einem Proxy-Programm spricht. Die Frage ist nun, wo der Code dafür abgelegt werden soll: Soll der Proxy in einer eigenständigen DLL liegen, oder wird er zusammen mit dem COM-Objekt selbst in der DLL abgelegt? Fürs Beispiel wählen Sie die Option "Allow merging...".

Die "MFC Support"-Option ist klar: Soll im Programm auch die MFC-Bibliothek verwendet werden können oder

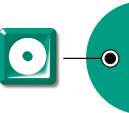


EIN NEUES ATL-OBJEKT fügen Sie per Objektmenü ein.

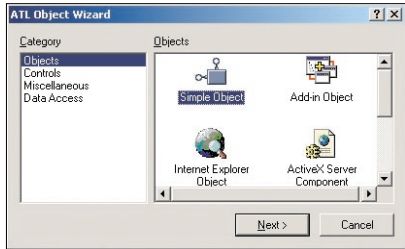
MFC-Projekten der Fall ist. Mit einem Klick auf die rechte Maustaste auf die Klassenansicht öffnen Sie das zugehörige Objektmenü und wählen darin den Befehl "New ATL Objekt".

Daraufhin erscheint der ATL-Objekt-Wizard, der eine ganze Reihe von unterschiedlichen Objektarten vorzuschlagen hat. Dazu muss man wissen, dass COM zwar ein allgemeingültiger Ansatz ist, dass das aber niemanden davon abhält, spezielle Interfaces bei COM-Objekten vorzusetzen.

So muss ein COM-Objekt, das vom IIS benutzt werden soll, einen bestimm-



ten Satz an Interfaces bzw. Methoden unterstützen – sonst kann es der IIS nicht laden. Welche Interfaces und Metho-

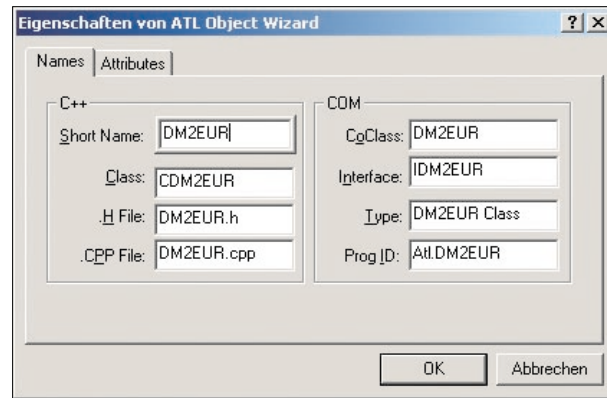


DER ATL-OBJEKT-WIZARD kennt eine ganze Reihe möglicher COM-Objekte.

den das sind, bestimmt die Anwendung, für die das Objekt geschrieben wird.

Nachdem Sie nur ein COM-Objekt programmieren wollen, aber keines, das den speziellen Anforderungen des IIS oder des MTS genügt, wählen Sie nur "Simple Object" aus der Liste der

Im folgenden Schritt müssen Sie die Eigenschaften Ihres Objektes festlegen. Das sind im Wesentlichen dessen Name, wobei der Wizard alle Namen aus dem unter "Short Name" eingegebenen Namen ableitet. Dabei wird zwischen

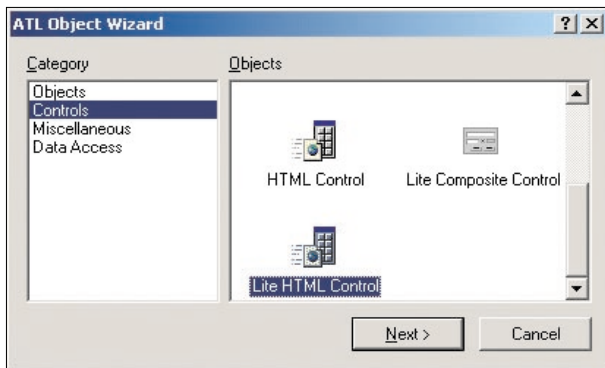


DIE NAMEN DES COM-OBJEKTES können mehr oder minder automatisiert vergeben werden.

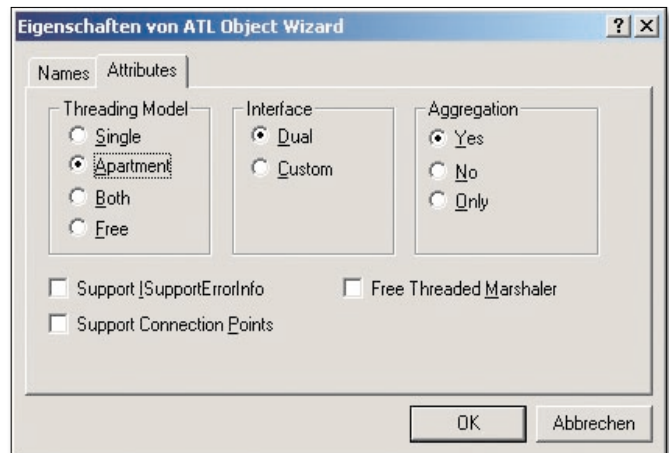
men diejenigen sind, die das Objekt in Ihrem C++-Projekt bei der Implementierung verwendet.

Sind die Namen vergeben, können Sie noch die Attribute für das Objekt einstellen. Für den allerersten COM-Versuch lassen Sie diese aber einfach auf den Vorgabewerten stehen.

Danach erscheint in der Klassenansicht im Arbeitsbereich schon deutlich mehr. Sie können hier verschiedene Symbole sehen: Zum einen gibt es ein Symbol, das sich aus mehreren Rechtecken zusammensetzt: Dahinter verbirgt sich die C++-Implementierung Ih-



ES GIBT AUCH SPEZIELLE ATL-COM-OBJEKTE für die Gestaltung von HTML-Controls.



vorgeschlagenen Objekte aus. (Sie können übrigens auch mehrere COM-Objekte in einer DLL ablegen, auch wenn das im Beispiel nicht ausdrücklich getan wird.)

den COM- und den C++-Namen unterschieden – die COM-Namen sind die, die vom COM-System aus zu verwenden sind, während die C++-Na-

DIE COM-ATTRIBUTE verändern Sie für das Beispiel nicht.

res Objekts, und im zugehörigen Ast tauchen nach und nach die C++-Methoden auf.

Darunter finden Sie ein Stecker-Symbol, und das symbolisiert das COM-Interface, das zu Ihrer C++-Implementierung gehört: Hier tauchen die COM entsprechenden COM-Methoden auf. Wenn Sie Ihre Klasse um eine einfache C++-Methode erweitern möchten, müssen Sie das per rechtem Mausklick aufs obere Symbol tun, während Sie mit

COM-OBJEKTE BENUTZEN

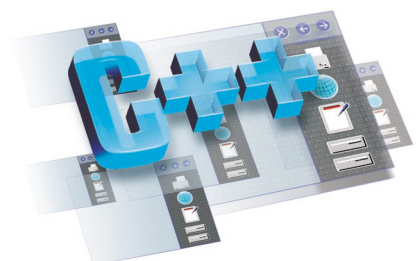
COM-Objekte können Sie praktisch von jeder Programmiersprache aus benutzen. In VC++ können Sie zum Beispiel die Komponenten-Galerie verwenden, um ein COM-Objekt auszuwählen. VC++ generiert dann den benötigten Wrapper-Code und gibt Ihnen damit die Möglichkeit, das COM-Objekt so zu benutzen, als wäre es einfach eine C++-Klasse.

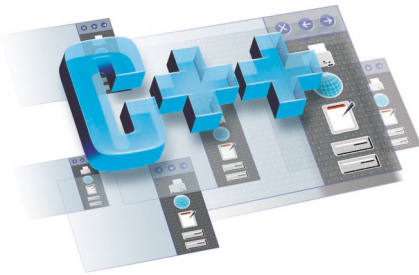
Auch in anderen Sprachen ist die Nutzung sehr einfach. Wenn Sie ein VBS-Script mit dem WSH ausführen, sähe die Verwendung des Beispiel-Objektes in etwa folgendermaßen aus:

Listing 3

```
set converter =  
Wscript.CreateObject("Atl2.DMEUR")  
dm = 12.5  
eur = 0.0  
converter.DM2EUR_VBS2 dm, eur  
Wscript.Echo(eur)
```

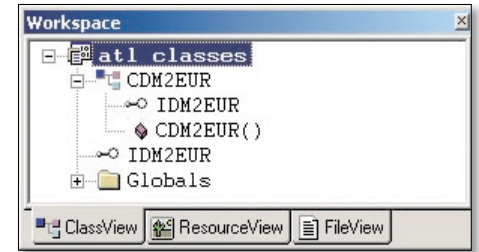
Wenn Sie sich nun das "Wscript" noch wegdenken und ein <script language=vbscript>-Tag außenherum denken, haben Sie schon die Variante zur Nutzung des Objekts in einer HTML-Seite oder in einem ASP-Skript. Wesentlich einfach geht es nicht.





einem rechten Mausklick aufs untere Symbol eine neue COM-Methode definieren, für die dann automatisch eine C++-Implementierung erzeugt wird.

Genau das tun Sie im folgenden Schritt: Das Objekt ist bereits definiert, aber es hat noch keine Methoden. Sie müssen also eine hinzufügen,



JETZT AUCH MIT EINEM COM-OBJEKT IM ARBEITSBEREICH: die Klassenansicht.

IDL, DLL UND DIE REGISTRY

Wie Sie im Beitrag erfahren haben, kümmert sich die COM-Library völlig transparent um das Laden von COM-Objekten und den Transport von Daten zwischen einem COM-Client und einem COM-Objekt. Damit das aber geht, muss die COM-Library natürlich ein gewisses Maß an Informationen über die Objekte, dessen Methode und deren Parameter haben.

Für COM-Objekte wird ein alter Windows-Mechanismus zum Transport von ausführbarem Code verwendet. Diese Objekte liegen meist in Form von DLLs vor. (Man kann COM-Objekte auch in .EXE-Dateien verpacken, das wird aber meist nicht getan.) Anders als bei normalen DLLs werden die COM-DLLs aber nicht direkt vom Windows-Program-Loader geladen, sondern von der COM-Library mit Hilfe des Program-Loaders. Dazu müssen die DLLs bestimmte Eintrittspunkte zur Verfügung stellen, mit denen COM die DLL nach ihrem Inhalt fragen kann. Diese werden allerdings beim Anlegen eines neuen COM-Objekts automatisch vom VC++-Assistenten erzeugt.

Außerdem muss das Objekt in der Windows-Registry registriert werden. Dabei wird in der Registry ein eindeutiger Identifizierer abgelegt, der auf die DLL-Datei verweist. Zusätzlich zu diesem eindeutigen Identifizierer gibt es auch noch einen von Menschen lesbaren Namen für ein COM-Objekt – den kann sich der Programmierer des Objektes aussuchen.

Soll das COM-Objekt nun geladen werden, wird dazu meist der von Menschen lesbare Name verwendet, denn schließlich schreibt ja auch ein Mensch den Quellcode, den das Objekt benutzen soll. Die COM-Library sucht nun anhand dieses Namens den zugehörigen eindeutigen Identifizierer in der Registry. Mit diesem kann die Library den Pfad zur DLL-Datei ermitteln, in dem das Objekt vorliegt.

Das bedeutet aber auch, dass all diese Informationen irgendwie in die Registry hereinkommen müssen. Praktischerweise wird das von VC++ erledigt. Dazu erzeugt

der Assistent beim Anlegen eines COM-Projektes eine .REG-Datei, die einfach in die Registry aufgenommen werden kann. Das geht zum Beispiel dadurch, dass man mit der rechten Maustaste auf die .REG-Datei klickt und dann den passenden Befehl auswählt. Allerdings ist dies nicht notwendig, da der Build-Prozess innerhalb von VC++ bei einem COM-Objekt um einen Schritt er-

tet, es handelt sich um eine Programmiersprache, die speziell für die Beschreibung von Objekten und deren Methoden und Parametern geschaffen wurde. Dabei enthalten IDL-Quellcodes keinerlei Anweisungen für einen Programmfluss oder Ähnliches, sondern ausschließlich Beschreibungen von Schnittstellen. Das sieht in etwa wie in Listing 4 aus.

LISTING 4

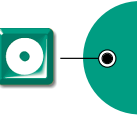
```
[
    object,
    uuid(1C15E1E1-36CD-11D6-8B65-0060972A8F90),
    dual,
    helpstring("IDMEUR-Schnittstelle"),
    pointer_default(unique)
]
interface IDMEUR : IDispatch
{
    [id(1), helpstring("Methode DM2EUR")] HRESULT
    ➤DM2EUR(double dm, double* eur);
    [id(2), helpstring("Methode EUR2DM")] HRESULT
    ➤EUR2DM(double eur, double* dm);
    [id(3), helpstring("Methode DM2EUR_VBS")] HRESULT
    ➤DM2EUR_VBS(VARIANT dm, VARIANT eur);
    [id(4), helpstring("Methode DM2EUR_VBS2")] HRESULT
    ➤DM2EUR_VBS2(VARIANT dm, VARIANT* eur);
};
```

weitert wurde: Nach dem Linken führt VC++ die Registrierung des Objektes automatisch durch.

Bleibt die Frage, wie die COM-Library nun ermitteln kann, wie die Datentypen der einzelnen Methoden im Objekt aussehen und wie andere Programme herausfinden können, welche Methode es überhaupt gibt.

Auch dazu müssen irgendwie Informationen vom Objekt zur Verfügung gestellt werden, und das passiert in Form von IDL- und .TLB-Dateien. IDL steht dabei für "Interface Definition Language" und TLB für "Type Library". IDL ist eine standardisierte Sprache zur Definition von Interfaces. Das bedeu-

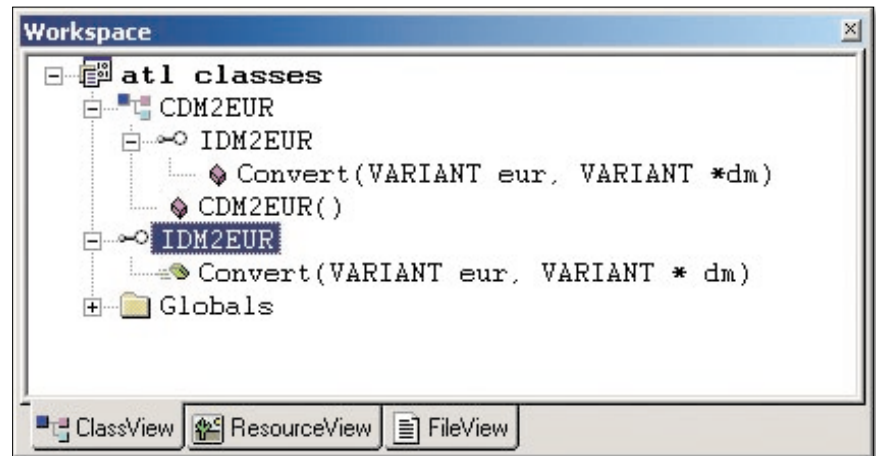
Man muss sich bei VC++-COM-Objekten nicht selbst um die Erstellung der IDL-Quellcodes kümmern: Auch das besorgen die Assistenten automatisch ohne weiteres zutun. Aus den IDL-Quellcodes wird nun wiederum eine Datei mit der Erweiterung .TBL erzeugt. Dabei handelt es sich um die "Type Library" und diese Library wird mit dem COM Objekt zusammen ausgeliefert. Die Type Library kann zum Beispiel mit in die DLL des Objektes gelinkt werden, oder sie wird als eigenständige Datei weitergegeben. Diese Type Library kann nun sowohl vom COM-System als auch von anderen Anwendungen benutzt werden, um Informationen über ein bestimmtes COM-Objekt zu erhalten.



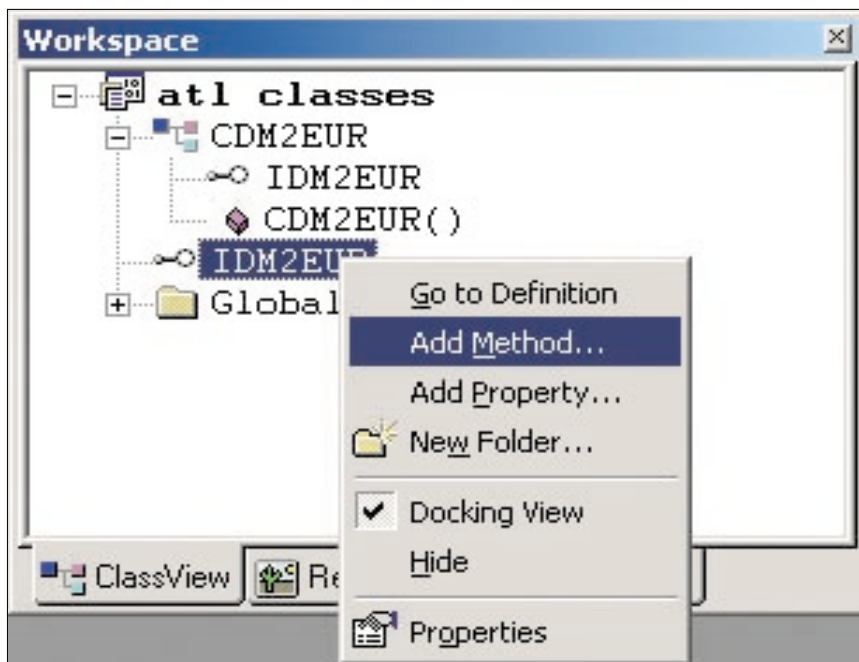
und das geht wie gesagt mit einem rechten Mausklick auf das COM-Objekt in der Klassenansicht.

Daraufhin öffnet sich der Dialog zum Hinzufügen einer Methode zu Ihrem Interface. Geben Sie der Methode einen Namen, und spezifizieren Sie dann die Parameter. Wie Sie bereits erfahren haben, benötigen Sie hier den Datentyp VARIANT, um Sprachen wie VBScript ebenfalls zu unterstützen. Die Parameter sind dabei einfach hintereinander und durch Komma getrennt einzugeben.

Wenn Sie nun den Dialog beenden, passiert Folgendes: Zum einen wird Ihr Interface mit der neuen Methode ausgestattet und zum anderen wird die C++-



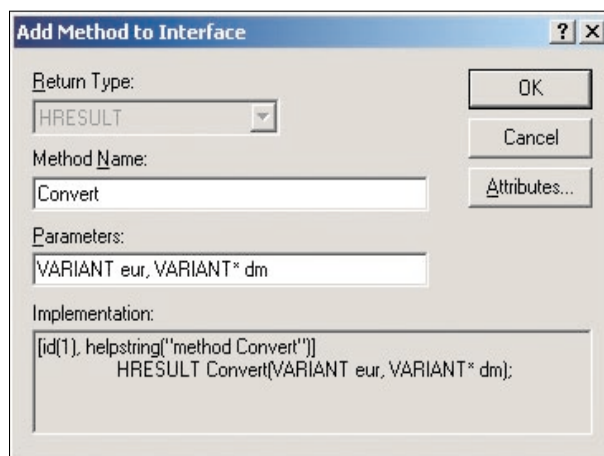
WENN SIE NUN AUF DEN NAMEN der C++-Methode doppelklicken (bei der Klasse, nicht beim Interface!) öffnet VC++ den Quelltexteditor und platziert sie an der richtigen Stelle.



MIT DIESEM BEFEHL erzeugen Sie eine neue Methode.

Implementierung Ihres Interfaces mit einer Default-Implementierung dieser Methode ausgestattet.

Die Funktion ist natürlich relativ leicht implementiert: Sie brauchen nur die Euro-Summe entsprechend zu dividieren oder die DM-Summe zu multiplizieren. Die einzige Frage, die sich stellt ist die, an welcher Stelle in der



BEIM HINZUFÜGEN EINER METHODE müssen Sie auch die Parameter mit angeben.

VARIANT-Struktur die Eingangsdaten zu finden sind, und an welche Stelle Sie die Ausgangsdaten schreiben müssen. Das ist am einfachsten mit dem Debugger herauszufinden.

Auf der Heft-CD zum Sonderheft finden Sie eine Beispiel-Implementierung, die mehrere Arten der *Convert()*-Funktion mit unterschiedlichen Parametern zum Ausprobieren bereitstellt.

Ferner finden Sie dort auch ein VBS-Skript, das das Beispiel-Objekt über den Windows Scripting Host benutzt und eine HTML-Seite, in der das Objekt ebenfalls benutzt wird. Diese beiden Dateien klären nun, wie Sie Ihre COM-Objekte von anderen Programmen aus benutzen. Die passenden Beispiel-Codes finden Sie im Kasten: "COM-Objekte benutzen".

In diesem Beitrag haben Sie erfahren, wie komplex COM ist – und wie einfach es mit ATL ist, ein wieder verwendbares Objekt zu implementieren, wenn man die Komplexität völlig ignoriert.

Trotzdem: Irgendwann hilft es nichts und Sie müssen sich auch mit den Details von COM auseinandersetzen – aber dafür haben Sie nun den richtigen Einstieg gefunden. ✓ UR

