Mit Stellenmarkt

4 199148 504504 02

€ 4,50 H 10554

iX

MAGAZIN FÜR PROFESSIONELLE INFORMATIONSTECHNIK

2 Februar 2002

Anti-Viren-Software für Firmennetze

E-Mail-Desinfektion

Getestet: 13 Produkte für IMAP und MS Exchange

Perl im Web:

Content-Management mit Mason

Datensicherung:

Backup-Strategien im Netz

Softwareentwicklung:

Komponenten für KDE

Objektorientiert und SQL-fähig:

Caché 4

Sicherheit:

Linux-Firewalls

Massenspeicher:

Tipps zur SCSI-Verkabelung

Administration:

Microsofts Operations Manager

Single Sign-In:

MS Passport und die Konkurrenz

Marktübersicht:

Revisionssichere Archivsysteme

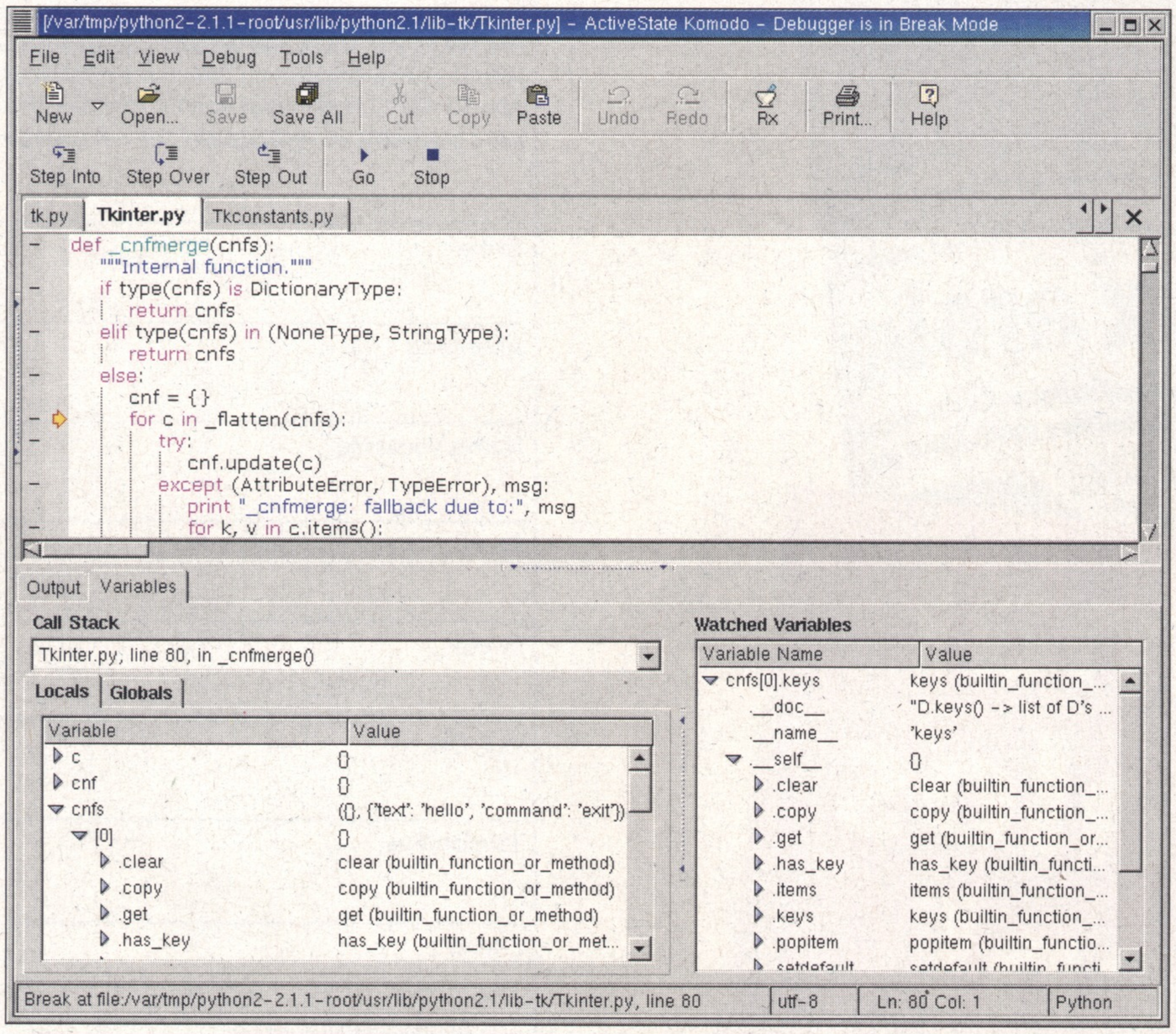
iX Magazin für professionelle Informationstechnik www.heise.de/ix/ Februar 2002 Österreich € 5,10 Schweiz str 8,80 Benelux € 5,20 HEISE

Komodo: IDE für Skriptsprachen

Viel hilft viel

Michael Schilli

Komodo, die integrierte Entwicklungsumgebung von *ActiveState*, bietet *Visual-Studio*-ähnlichen Komfort für eine Palette von Skriptsprachen. Ein Praxistest des Produkts im rauen Programmieralltag.



Statt eines simplen Editors und eines Compilers oder Interpreters, der von der Kommandozeile startet, bevorzugen viele Entwickler eine integrierte Entwicklungsumgebung (Integrated Development Environment, IDE). Diese kennt alle Dateien eines Projekts und kann schnell zwischen verwandten Programm- und Header-Dateien hin- und herspringen. Sie übersetzt, linkt oder interpretiert und lässt das Programm im Ausgabe- oder Debugger-Fenster ablaufen.

Typische IDE-Produkte heben Schlüsselwörter des Programmtexts sprachspezifisch farblich hervor (Syntax-Highlighting) und zeigen manchmal sogar Syntaxfehler an, die sich beim Eintippen einschleichen, da der eingebaute Editor entweder die gerade programmierte Sprache versteht oder im Hintergrund schon ein Compile-Lauf stattfindet.

Echte Programmierer fahren jedoch kaum mit diesen S-Klasse-Mercedes der Softwareentwicklung: Wahre Hacker bevorzugen schnelle und gefährliche Editoren wie *vi*, können *make*-Regeln definieren und wissen auswendig, wie man in CVS automatisch zwei Zweige eines Projekts konsolidiert.

ActiveState beschreitet mit *Komodo* neues Terrain und bietet eine IDE für Skripthacker an. Es kommt gegenwärtig mit Perl, PHP, Python, JavaScript, Tcl und XSLT zurecht. Das kommerzielle Produkt basiert auf dem frei erhältlichen Mozilla-Code, definiert seine Oberfläche mittels *XUL* (XML-based user interface language) und steht unter www.activestate.com/Products/Komodo zum Download bereit.

Mehrere Sprachen in einem Projekt

Die Linux-Version lag im Test als Beta-Release 1.1 vor und ist frei erhältlich. Für die Windows-Plattform bietet ActiveState drei Wahlmöglichkeiten: Die 'kommerzielle' Lizenz für 295 US-$, die kostenlose Evaluierungslizenz für 21 Tage und die Privatnutzerlizenz für 29,50 US-$, die den uneingeschränkten Gebrauch für nicht kommerzielle Zwecke anbietet.

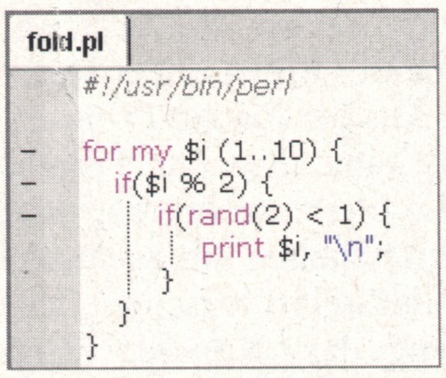
Sowohl unter Linux als auch unter Windows sorgen Installationsprogramme für die reibungslose Einrichtung des Produkts. Beide Varianten kommen vollständig an, setzen also keinen installierten Mozilla-Browser voraus und bieten die gleichen Funktionen. Unterschiede liegen lediglich darin, dass unter Linux die unterstützten Skriptinterpreter meist schon von Haus aus installiert sind (zum Beispiel *python*), während Windows-Anwender sie nachrüsten müssen.

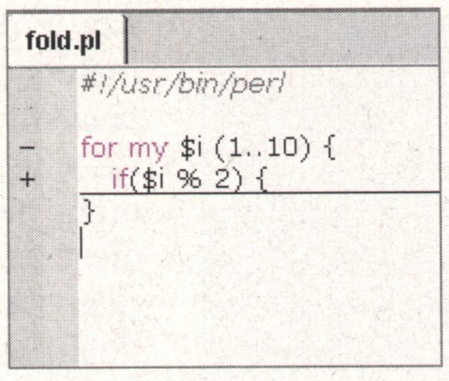
Nach dem Start zeigt Komodo beispielhaft, dass ein Projekt eine Reihe von Dateien aller unterstützten Sprachen enthalten darf – allerdings nur in einer Hierarchiestufe, ohne die Möglichkeit, sie auf Unterverzeichnisse zu verteilen. Die IDE merkt sich die zuletzt editierten Dateien und Projekte, was die schnelle Rückkehr zum aktuellen Stand gewährleistet. Die neue Version 1.2 merkt sich sogar, wo genau man die Arbeit beendete.

Syntaxprüfung beim Tippen

Während des Tippens unterlegt die IDE Schlüsselwörter mit verschiedenen Farben, was die Lesbarkeit erhöht. Obwohl Perls komplexe Syntax eigentlich nur sein eigener Interpreter komplett versteht, da die Bedeutung vieler Konstrukte erst feststeht, wenn der Kontext bekannt ist, löst Komodo diese Aufgabe überzeugend.

Klickt man nahe einer sich öffnenden Codeblock-Klammer ({}) auf ein Minus-Symbol im Seitenbalken, lässt der Editor den Block zusammenschrumpfen (Abbildung 1) – nützlich, um sich einen Überblick über große Dateien zu verschaffen.





Code mit komplettem (links) und kollabiertem *if*-Statement (rechts, Abb. 1)

Aber Komodos Syntaxverständnis geht noch über die üblichen Farbmarkierungen hinaus. Für Perl, PHP und Python (für Tcl nur, falls es eine TclPro-Installation findet) wirft es im Hintergrund den passenden Interpreter an und lässt ihn das Skript während der Eingabe analysieren. Meldet er Fehler oder Warnungen, greift der Editor diese auf und unterringelt die betroffenen Zeilen rot oder grün. Fährt der Programmierer mit der Maus über die Zeile, erscheint der Fehler- oder Warnungstext.

Beim Erzeugen von Perl-Code schaltet Komodo den Warnungsmodus vernünftigerweise selbst dann ein, wenn das Skript nicht mit *-w* oder *use warnings* danach verlangt; diese Vorsichtsmaßnahme ist abstellbar.

Verlangt der Benutzer nach *use strict*, kommen weitere sinnvolle Warnungen hinzu, etwa im Fall einer nur im Schleifenrumpf definierten Variablen, die versehentlich außerhalb verwendet wird. Da der Perl-Interpreter im *use strict*-Modus auf solche Leichtsinnsfehler anspringt, unterringelt Komodo die auslösende Zeile grün. Außerdem bietet der Editor einige andere Schmankerln, etwa das Auskommentieren selektierter Bereiche per Menüeintrag.

Fehlersuche nah und fern

Perl liefert einen Debugger für die Kommandozeile mit, auf den Komodo zur einfacheren Bedienung besonders für Neulinge eine Oberfläche setzt. Im Variablenfenster ('Proximity') erscheinen die im aktuellen Bereich gültigen Datenstrukturen des Programms. Von tief verschachtelten Strukturen zeigt Komodo zunächst nur die oberste Ebene an. Klickt man mit der Maus aber auf den grünen Pfeil, der zur Tiefenbohrung einlädt, kann man die Datenstruktur bis in beliebige Tiefen sehen (siehe Abbildung S. 64).

Variablen lassen sich per Drag and Drop in das rechts davon gelegene Watch-Fenster ziehen und ihre Änderung bei jeder ausgeführten Codezeile beobachten. Benutzt ein Perlskript Module, die ihrerseits wieder Module anfordern, sind schnell mal fünf Dateien gleichzeitig aktiv. Die Reiter über dem Codefenster zeigen die bisher geladenen Dateien an und ermuntern dazu, mit einem Mausklick den dort stehenden Code aufzublättern.

Verfügt der Rechner, auf dem das zu analysierende Skript läuft, über keine Komodo-Installation oder gar kein Terminal, kann ein anderswo laufendes Komodo das Debugging übernehmen. Für Perl wird hierzu einfach ein der Distribution beiliegendes Skript auf die Remote-Maschine kopiert. Ein Menüklick weist das laufende Komodo an, auf einem festgelegten Port auf Signale des ferngesteuerten Skripts zu lauschen. Startet man Letzteres anschließend auf der Remote-Maschine, sorgen das Debugger-Skript und Umgebungsvariablen dafür, dass es als erstes Kontakt zur fernsteuernden Maschine aufnimmt. Dabei spielt es keine Rolle, ob Komodo- und Remote-Maschine unterschiedliche Betriebssysteme fahren, wie Abbildung 3 zeigt.

Perls reguläre Ausdrücke erschlagen Anfänger oft mit ihrer Komplexität. Der *Rx* genannte Debugger soll Komodo-Anwendern die Arbeit erleichtern. Anhand eines Beispieltexts zeigt er, welche Zeichen ein regulärer Ausdruck erfasst und was er mittels der greifenden Klammern *(…)* einfängt. Spielt man am regulären Ausdruck, ändert sich die farbliche Unterlegung im Beispieltext in Echtzeit. Die Implementierung dieser Funktion lässt allerdings stark zu wünschen übrig. Während *Rx* geklammerte Treffer zuverlässig hervorhebt, zeigt der mitgelieferte Debugger, der die Regex-Maschine schrittweise durch den Beispieltext führen soll, teilweise stark verwirrende Daten an. *Rx* basiert auf einer guten Idee, die Realisierung in Komodo bedarf aber einer Überarbeitung.

Python- oder Tcl-Anfängern (zukünftig auch Perl-Neulingen) schlägt der Komodo-Editor in den aktuellen Kontext passende Befehle vor und gibt Nachhilfe, was deren Syntax betrifft. Fällt dem Programmierer gerade nicht ein, dass die *puts*-Funktion in Tcl zwei optionale Argumente gefolgt vom auszugebenden Textstring erwartet, hilft die aufleuchtende Gedächtnisstütze weiter. In Abbildung 4 hat der Entwickler *set value [str…* getippt und Komodo weiß, dass dies der Anfang einer von Tcls *string*-Funktionen sein könnte.

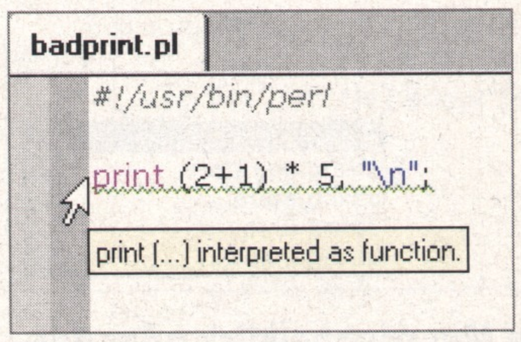
XSL-Transformationen live verfolgen

Da nicht jeder mächtige Programmiersprachen wie Perl beherrscht, um XML-Dokumente zu transformieren (etwa in HTML), gibt es XSLT-Stylesheets [1]. Mit einer XML-basierten Sprache legt man dort fest, wie der XSLT-Prozessor die Elemente der XML-Datei umwandeln muss, um beim Zielformat anzukommen.

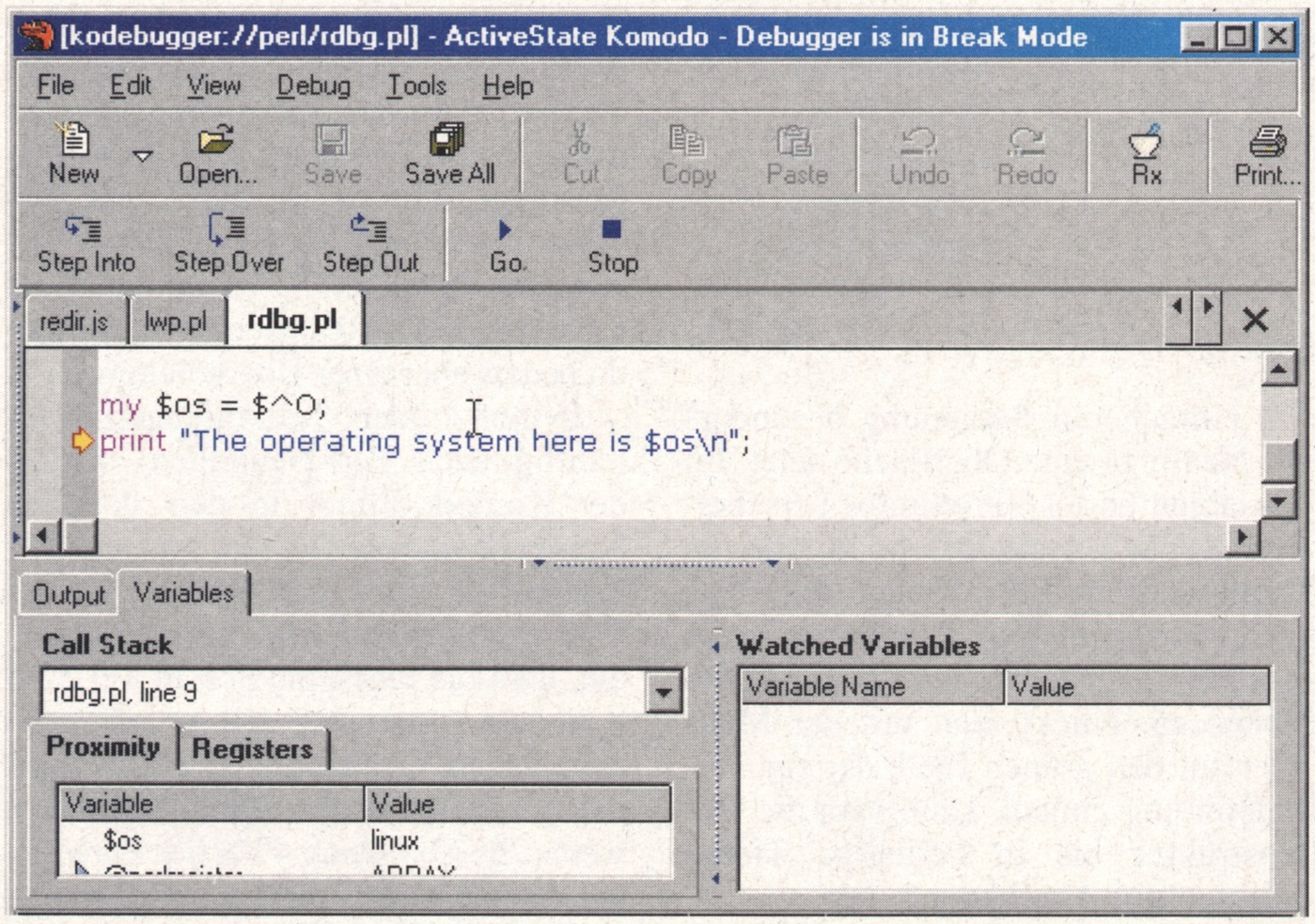
Komodo bietet einen Debugger für den XSLT-Prozessor, der sich schrittweise durch das XML-Dokument und das XSLT-Stylesheet arbeitet und im Ausgabefenster mit jedem durchschrittenen XML-Knoten und jeder abgearbeiteten XSLT-Regel häppchenweise das Zielformat ausgibt. Zusammen mit der XML-Tag-Komplettierung erleichtert dies das Erstellen von XML-Dokumenten und XSLT-Stylesheets erheblich. Wer gerade XSLT lernt, dem sei der Komodo-Debugger wärmstens empfohlen. Während statisch definierte Regeln manchmal schwer zu verdauen sind, hilft

iX-TRACT

- Komodo ist eine auf Mozilla basierende Entwicklungsumgebung für Skriptsprachen, es läuft unter Windows und Linux.
- PHP-, Perl- und Python-Code prüft die IDE beim Eingeben auf Syntaxfehler und markiert diese gegebenenfalls.
- Schrittweises Ausführen von XSLT-Skripts hilft beim Verstehen der von ihnen ausgeführten Transformationen von XML in HTML.



Komodo erkennt den berüchtigten Anfängerfehler: ein falscher Aufruf von Perls *print*-Funktion (Abb. 2)



Ein Komodo-Debugger unter Windows schreitet durch ein Perlskript auf einem Linux-Rechner (Abb. 3).

es sehr, dem Prozessor über die Schulter zu schauen, um zu verfolgen, was tatsächlich zur Laufzeit passiert. Diese Funktion allein rechtfertigt es, einen genaueren Blick auf das Produkt zu werfen.

Für eine Reihe weiterer Sprachen bietet Komodo zumindest auf Schlüsselwörtern basierende Syntax-Hervorhebung an. Dabei unterlegt es allerdings stur bestimmte Zeichenfolgen, es steckt keine wirkliche Intelligenz dahinter. Unterstützt werden unter anderem LaTeX, C++ und Java, aber wer für diese Sprachen professionelle IDEs sucht, ist mit spezialisierteren Produkten besser bedient.

Auf wackligen Füßen

Größte Schwäche des Produkts ist seine geringe Stabilität. Die grafische Oberfläche der als Beta vorliegenden Linux-Version stieg bei schnellen Mausaktionen aus und zwang häufig zu Neustarts. Die kurz nach Ende des Tests freigegebene Version 1.2 ließ sich auf dem Referenzrechner nicht installieren. Auch mit plötzlichen Core-Dumps musste man leben. Selbst die von ActiveState als stabil bezeichnete Vollversion für Windows zeigte Mängel, falls der analysierte Code komplexer als die mitgelieferten Beispiel-Snippets wird. Steigt man bei folgendem Perl-Code schrittweise im Debugger in die Tiefen der eingebundenen Module, reduziert sich das Tempo drastisch:

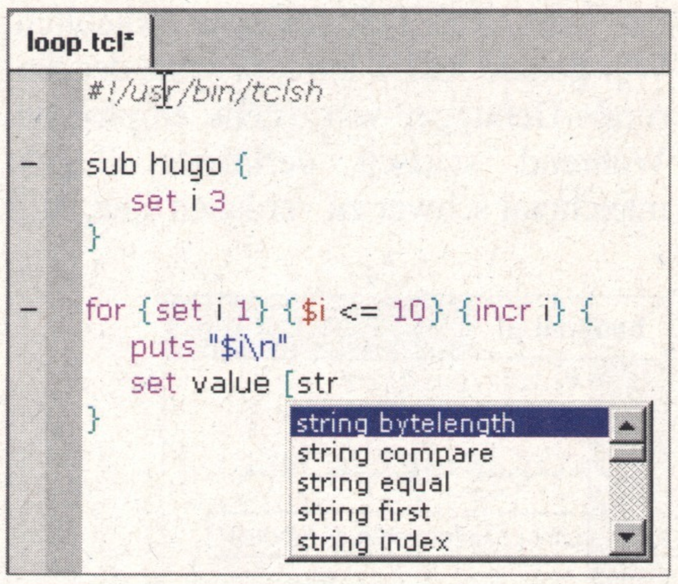
```
use Net::FTP;
my $ftp = Net::FTP->new("some.host.com");
```

Wenn jeder Einzelschritt die Oberfläche für zehn Sekunden einfriert, verzichtet man gern auf den Komfort eines grafischen Debuggers und arbeitet lieber von der Kommandozeile aus. In der Version 1.2 arbeitet der Debugger jedoch deutlich schneller.

ActiveState hatte offenbar Version 1.1 nur rudimentär getestet. Es zeigen sich teilweise haarsträubende Fehler: Ruft man einmal den Debugger für eine Datei auf, kann man nicht mehr zurück zur Liste der Projektdateien. In der Beta 1.2 war dieser Fehler behoben. Legt man mit dem GUI ein neues Projekt an und löscht anschließend die dazugehörige **.kpf*-Datei von Hand, liegt *komodo* wie eine Schildkröte auf dem Rücken. Es zeigt nach wie vor das Projekt in *Recent Projects* an, meldet aber beim Öffnen einen *unexpected error*. Aus der *Recent Projects*-Liste löschen kann man das Projekt nicht mehr. Es bleibt zu hoffen, dass diese Kinderkrankheiten mit wachsendem Anwenderkreis verschwinden. Positiv fiel auf, dass Komodo seine Daten (wie die Projektdatei) in XML ablegt – so kann man notfalls von Hand eingreifen.

Es bot sich an, die Dokumentation als HTML mitzuliefern und auf Knopfdruck mit Mozilla anzuzeigen. Zu finden sind jedoch lediglich die Beschreibungen aller verfügbaren Menüs sowie einige Tutorial-ähnliche Anweisungen zur Bedienung der verschiedenen Applikationen. Mehr wäre mehr gewesen. Laut Herstellerangaben ist die Dokumentation in Version 1.2 verbessert.

ActiveState hat viele unterschiedliche Funktionen in ein einzelnes Produkt verpackt, um es einem möglichst großen Anwenderkreis schmackhaft zu machen. Es dürfte eine große Herausforderung werden, diese Einzelteile stabil zu halten. Eine Aufgabe, die die Open-Source-Gemeinde auf Anfrage sicher dankbar aufgriffe. (ck)



Auf einen Mausdoppelklick oder das Drücken der Tab-Taste hin übernimmt der Editor die vorgeschlagene Funktion in den Code (Abb. 4).

iX-WERTUNG

⊕ Unterstützung für viele Skriptsprachen
⊕ Syntaxprüfung während der Eingabe
⊕ Remote-Debugging möglich
⊕ einzigartiger XSLT-Debugger

⊖ instabil
⊖ Debugger zu langsam

MICHAEL SCHILLI

arbeitet als Web-Engineer für AOL/Netscape in Mountain View, Kalifornien. Er hat 'Goto Perl 5' (deutsch) und 'Perl Power' (englisch) für Addison-Wesley geschrieben.

Literatur

[1] Henning Behme; XML-Programmierung; Mutabor; XSLT-Tutorial I-III; *iX* 1/2001, S. 167; 2/2001, S. 142; 3/2001, S. 167