

# Der Editor Vim/gVim als Perl-IDE

Fritz Mehner

Fachhochschule Südwestfalen  
Iserlohn

11. Deutscher Perl-Workshop  
Frankfurt am Main  
25.-27.02.2009

# Inhalt

IDEs und Programmiereditoren

Das Plug-in Perl Support

Weitere Plug-ins

Ausblick – Was fehlt?

# IDEs und Programmiereditoren

## Ein paar Gesichtspunkte

EDITOR	IDE
<ul style="list-style-type: none"><li>+ umfangreiche Editierfunktionen</li><li>+ viele Programmiersprachen</li></ul>	<ul style="list-style-type: none"><li>+ Klassenbrowser</li><li>+ Projektverwaltung</li><li>+ Debugger</li><li>+ Dokumentation</li><li>+ Bibliotheken</li><li>+ leistungsfähige GUI</li></ul>
<ul style="list-style-type: none"><li>- GUI oft eingeschränkt</li><li>- wichtige Leistungsmerkmale fehlen</li></ul>	<ul style="list-style-type: none"><li>- wenige Programmiersprachen</li><li>-- Editor mäßig</li></ul>

# Das Plug-in Perl Support

## Leistungsmerkmale

- Einfügen von vollständigen Anweisungen, Kommentaren, Idioms, Code-Schnipseln und POD-Anweisungen
- Menüs (gVim), Hotkeys (Vim, gVim)
- vollständige Perl-Dokumentation erreichbar
- regulärer Ausdrücke : zusammensetzen, analysieren, testen
- Skriptausführung, Syntaxprüfung
- `perltidy`, `perlritic`, `podchecker`, `pod2xxx` ausführen
- Debugger, Profiler starten
- Template-System
- ...

# Abhängigkeiten

Modul	Beschreibung	Anmerk.
Pod::Checker	check pod documents for syntax errors	
Pod::Html	module to convert pod files to HTML	
Pod::Man	convert POD data to formatted *roff input	
Pod::Perldoc	display Perl documentation	
Pod::Text	convert POD data to formatted ASCII text	
Devel::SmallProf	per-line Perl profiler	opt.
Perl::Critic	critique Perl source code for best-practices	opt.
Perl::Tags	Ctags style tags for Perl sourcecode	opt.*
Perl::Tidy	parses and beautifies perl source	opt.
YAPE::Regex::Explain	explanation of a regular expression	opt.*

\* *benötigt die Vim-Perl-Schnittstelle*

# Kommentare, Anweisungen, Code-Schnipsel, ...

Perl
Comments
Statements
Idioms
Snippets
Regex
File-Tests
Spec_Yar
POD
Run
help (plugin)

Comments	Perl
	end-of-line com.
	adjust end-of-line com.
	set end-of-line com. col.
	frame comm.
	function descr.
	method descr.
	file header (.pl)
	file header (.pm)
	file header (.t)
	file header (.pod)
	toggle comment
	comment block
	uncomment block
	date
	date time
	KEYWORD+comm.
	tags (plugin)
	vim modeline

Idioms	Perl
	my \$;
	my \$ = ;
	my ( \$, \$ );
	my @;
	my @ = (,); (1)
	my %;
	my % = (=>); (2)
	my \$rgx_ = q//; (3)
	\$ =~ m{ }xm
	\$ =~ s{ }xm
	\$ =~ tr{ }xm
	subroutine
	print "...n";
	open input file
	open output file
	open pipe
	<STDIN>
	<STDOUT>
	<STDERR>

Statements	Perl
	do {} while
	for {}
	foreach {}
	if {}
	if {} else {}
	elsif {}
	unless {}
	unless {} else {}
	until {}
	while {}
	{ }

Run	Perl
update_run_script	<C-F9>
update_check_syntax	<A-F9>
cmd_line_arg	<S-F9>
perl_switches	
start_debugger	<F9>
make_script_executable	
read_perldoc	<S-F1>
show_installed_Perl_modules	
generate_Perl_module_list	
run_peritidy	
run_SmallProf	
run_perlcritic	
perlcritic_severity (1)	▶
perlcritic_verbosity	▶
save_buffer_with_timestamp	
hardcopy_to_FILENAME.ps	
settings_and_hot_keys	
xterm_size	
output_VIM->buffer->xterm	

# Template-Definitionen (Datei Templates, Auszug)

```

$ ===== USER MACROS =====
$
|AUTHOR|      = Dr. Fritz Mehner
|AUTHORREF|   = mn
|EMAIL|       = mehner@fh-swf.de
|COMPANY|     = FH Südwestfalen, Iserlohn
|COPYRIGHT|   = Copyright (c) |YEAR|, |AUTHOR|
$
$ ===== FILE INCLUDES =====
$
|includefile| = file-description.template
$
$#####
== comment.keyword-todo == append ==
# :TODO:|DATE| |TIME|:|AUTHORREF|: <CURSOR>
== comment.keyword-workaround == append ==
# :WORKAROUND:|DATE| |TIME|:|AUTHORREF|: <CURSOR>
== comment.keyword-keyword == append ==
# :|?KEYWORD:u|:|DATE| |TIME|:|AUTHORREF|: <CURSOR>
$#####
== statements.do-while == below ==
do {
<SPLIT> } while ( <CURSOR> );           # ---- end do-while ----
$#####
== idioms.subroutine == below ==
sub |?FUNCTION_NAME| {
    my ( $par1<CURSOR> ) = @_;
<SPLIT> return ;
} # ----- end of subroutine |FUNCTION_NAME| -----

```

# Hotkeys

VIM-PLUG-INS  
perl-support.vim  
VERSION 4.1

## HOT KEYS

Key mappings for Vim and gVim.  
Plugin: <http://vim.sourceforge.net>  
Fritz Mehner ([mehner@fhwz.de](mailto:mehner@fhwz.de))

(i) insert mode, (n) normal mode, (v) visual mode

<i>Load / Unload Perl Support</i>	
<b>\lps</b>	load menus (n)
<b>\ups</b>	unload menus (n)
<i>Help</i>	
<b>\hp</b>	help (plugin) (n)
<i>Comments</i>	
<b>\cl</b>	end-of-line comment (n, v, i)
<b>\cj</b>	adjust end-of-line comments (n, v, i)
<b>\cs</b>	set end-of-line comment col. (n)
<b>\cfr</b>	frame comment (n, i)
<b>\cfu</b>	function description (n, i)
<b>\cm</b>	module description (n, i)
<b>\ch</b>	file header (.pl) (n)
<b>\ce</b>	file header (.pm) (n)
<b>\ca</b>	file header (.t) (n)
<b>\cp</b>	file header (.pod) (n)
<b>\ckb</b>	keyword comm. BUG (n, i)
<b>\ckt</b>	keyword comm. TODO (n, i)
<b>\clr</b>	keyword comm. TRICKY (n, i)
<b>\clw</b>	keyword comm. WARNING (n, i)
<b>\cso</b>	keyword comm. WORKAROUND (n, i)
<b>\ckn</b>	keyword comm. new keyword (n, i)
<b>\cc</b>	code ↔ comment (n, v)
<b>\cb</b>	code block → comment (n, v)
<b>\cn</b>	uncomment code block (n)
<b>\cd</b>	date (n, i)
<b>\ct</b>	date & time (n, i)
<b>\cv</b>	vim modeline (n, i)

<i>Statements</i>	
<b>\sd</b>	do { } while (n, v, i)
<b>\sf</b>	for { } (n, v, i)
<b>\sfe</b>	foreach { } (n, v, i)
<b>\sei</b>	elsif { } (n, v, i)
<b>\si</b>	if { } (n, v, i)
<b>\sie</b>	if { } else { } (n, v, i)
<b>\su</b>	unless { } (n, v, i)
<b>\sue</b>	unless { } else { } (n, v, i)
<b>\st</b>	until { } (n, v, i)
<b>\sw</b>	while { } (n, v, i)
<b>\s{</b>	{ } (n, v, i)
<i>Snippet</i>	
<b>\nr</b>	read code snippet (n)
<b>\nw</b>	write code snippet (n, v)
<b>\nw</b>	edit code snippet (n)
<b>\ntl</b>	edit local templates (n)
<b>\ntg</b>	edit global templates (n)
<b>\ntr</b>	reveal the templates (n)
<i>Regular Expressions</i>	
<b>\xr</b>	pick up RegEx (n, v)
<b>\xs</b>	pick up string (n, v)
<b>\xf</b>	pick up flag(s) (n, v)
<b>\xm</b>	match (n)
<b>\xe</b>	explain RegEx (n, v)
<i>Literals</i>	
<b>\\$</b>	my \$; (n, i)
<b>\\$=</b>	my \$ = ; (n, i)
<b>\\$\$</b>	my ( \$, \$ ); (n, i)
<b>\@</b>	my @; (n, i)
<b>\@=</b>	my @ = (,,); (n, i)
<b>\%</b>	my %; (n, i)
<b>\%=</b>	my % = (=>=>); (n, i)
<b>\ir</b>	my \$rgpc = q//; (n, i)
<b>\im</b>	\$ => m//xm (n, i)
<b>\is</b>	\$ => s//xm (n, i)
<b>\it</b>	\$ => tr//xm (n, i)
<b>\isu</b>	Subroutine (n, v, i)
<b>\ip</b>	print "...\n!"; (n, i)
<b>\io</b>	open input file (n, v, i)
<b>\io</b>	open output file (n, v, i)
<b>\ipi</b>	open pipe (n, v, i)

<i>POSIX Character Classes</i>	
<b>\pa</b>	[:alnum:] (n, i)
<b>\ph</b>	[:alpha:] (n, i)
<b>\pi</b>	[:ascii:] (n, i)
<b>\pb</b>	[:blank:] (n, i)
<b>\pc</b>	[:cntrl:] (n, i)
<b>\pd</b>	[:digit:] (n, i)
<b>\pg</b>	[:graph:] (n, i)
<b>\pl</b>	[:lower:] (n, i)
<b>\pp</b>	[:print:] (n, i)
<b>\pn</b>	[:punct:] (n, i)
<b>\ps</b>	[:space:] (n, i)
<b>\pu</b>	[:upper:] (n, i)
<b>\pw</b>	[:word:] (n, i)
<b>\px</b>	[:xdigit:] (n, i)
<i>Run</i>	
<b>\rr</b>	update file, run script (n)
<b>\rs</b>	update file, check syntax (n)
<b>\ra</b>	set command line arguments (n)
<b>\rw</b>	set Perl cmd. line switches (n)
<b>\rd</b>	start debugger (n)
<b>\re</b>	make script executable (n)
<b>\rp \h</b>	read perldoc for word under cursor (n)
<b>\ri</b>	show installed Perl modules (n)
<b>\rg</b>	generate Perl module list (n)
<b>\ry</b>	run perltidy (n, v)
<b>\rm</b>	run SmallProf (n)
<b>\rc</b>	run perlcritic (n)
<b>\rt</b>	save buffer with timestamp (n)
<b>\rh</b>	hardcopy buffer (n, v)
<b>\rk</b>	settings and hotkeys (n)
<b>\rx</b>	set xterm size (n, ctrl+o)
<b>\ro</b>	change output destination (n, v)

Ex commands for perlcritic (version 1.01+):

:CriticSeverity 1 2 3 4 5  
                  bunat cruel harsh stern gentle

:CriticVerbosity 1...11

:CriticOptions option(s), see perlcritic(1)

Ex command for Devel::SmallProf:

:SmallProfSort line number|line count|time|ctime

# Regulären Ausdruck erläutern

regulären Ausdruck markieren, *explain regex* aufrufen:

```

regex-test.pl (+) - GVIM
Datei Editieren Werkzeuge Syntax Puffer Ansicht Netzw Perl Filter Hilfe
The regular expression:
(?-imsx:([\w\s]+)(jumped){1,6})([\w\s]+)
matches as follows:
NODE                EXPLANATION
-----
(?-imsx:            group, but do not capture (case-sensitive)
                    (with ^ and $ matching normally) (with . not
                    matching \n) (matching whitespace and #
                    normally):
(                  group and capture to \1:
  [\w\s]+         any character of: word characters (a-z,
                    A-Z, 0-9, _), whitespace (\n, \r, \t,
                    \f, and " ") (1 or more times (matching
                    the most amount possible))
)                end of \1
REGEX-EXPLAIN      16,1      Top
([\w\s]+)(jumped){1,6})([\w\s]+)
### The quick brown fox jumped over the lazy dog. ###
regex-test.pl [+ ] 26,0-1    7%

```

# Regular Expression Tester

regulären Ausdruck markieren, Text markieren, *match* aufrufen:

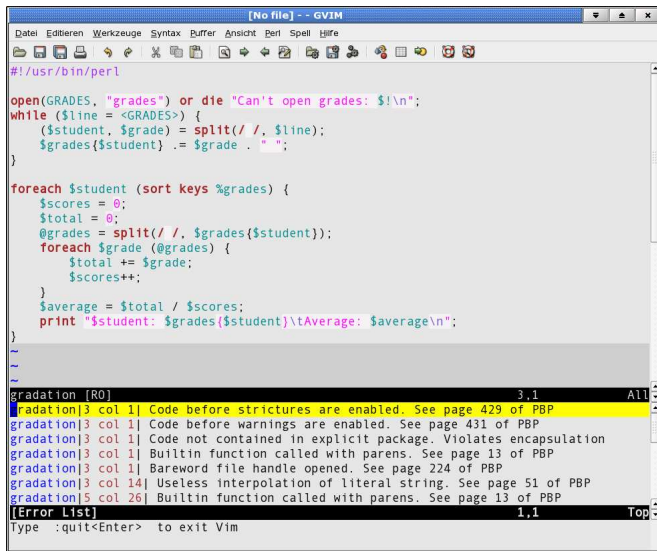
```
regex-test.pl+ (-) - Gvim
Datei Editieren Werkzeuge Syntax Puffer Ansicht Netzw Perl Filter Hilfe
REGEXP = m{([\w\s]+)(jumped){1,6}([\w\s]+)}
STRING [ 0, 53] = ### The quick brown fox jumped over the lazy dog. ###
prematch [ 0, 3] = ###
MATCH [ 3, 45] = | The quick brown fox jumped over the lazy dog|
postmatch [ 48, 5] = . ###

$1 [ 3, 21] = | The quick brown fox |
$2 [ 24, 6] = jumped
$3 [ 30, 6] = | over |
$4 [ 36, 12] = the lazy dog

$+ [ 12] = (the lazy dog)
$^N [ 12] = (the lazy dog)

REGEX-TEST 1,1 All
([\w\s]+)(jumped){1,6}([\w\s]+)
### The quick brown fox jumped over the lazy dog. ###
regex-test.pl [+ 29, 0-1 7%
```

# Integration von perlritic



The screenshot shows a GVIM window with a Perl script. The script calculates the average grade for each student from a file named 'grades'. The code uses 'split' to parse the file and 'foreach' loops to calculate the average. The error list at the bottom shows several warnings from Perlritic, such as 'Code before strictures are enabled' and 'Code not contained in explicit package'.

```
[No file] -- GVIM
Datei Editieren Werkzeuge Syntax Buffer Ansicht Perl Spell Hilfe
#!/usr/bin/perl

open(GRADES, "grades") or die "Can't open grades: $!\n";
while ($line = <GRADES>) {
    ($student, $grade) = split(/ /, $line);
    $grades{$student} .= $grade . " ";
}

foreach $student (sort keys %grades) {
    $scores = 0;
    $total = 0;
    @grades = split(/ /, $grades{$student});
    foreach $grade (@grades) {
        $total += $grade;
        $scores++;
    }
    $average = $total / $scores;
    print "$student: $grades{$student}\tAverage: $average\n";
}
~
~
~
gradation [R0] 3.1 All
gradation]3 col 1| Code before strictures are enabled. See page 429 of PBP
gradation]3 col 1| Code not contained in explicit package. Violates encapsulation
gradation]3 col 1| Builtin function called with parens. See page 13 of PBP
gradation]3 col 1| Bareword file handle opened. See page 224 of PBP
gradation]3 col 14| Useless interpolation of literal string. See page 51 of PBP
gradation]5 col 26| Builtin function called with parens. See page 13 of PBP
[Error List] 1.1 Top
Type :quit<Enter> to exit Vim
```

# Integration von Devel::SmallProf

The screenshot shows a Perl IDE window with a menu bar (Datei, Editieren, Werkzeuge, Syntax, Puffer, Ansicht, Perl, Filter, Hilfe) and a toolbar. The main editor area contains Perl code for a Sudoku solver. Below the code, a profiler output is displayed, showing execution time for various lines of code. The output is as follows:

```

#-----
my $smi = $row - $row % 3;
my $smj = $col - $col % 3;
foreach my $i ( $smi .. ( $smi + 2 ) ) {
    foreach my $j ( $smj .. ( $smj + 2 ) ) {
        found[ $sudoku_ref->[$i][$j] ]++;
    }
}

#-----
# RESTRICTIONS
# check 1. diagonal (if requested)
Solver.pm [R0] 198,17 24%
Solver.pm|226| 39789:6:89: if ( $found[$i] == 0 ) {
Solver.pm|198| 39789:10:120: $found[ $sudoku_ref->[$i][$
Solver.pm|188| 39789:10:189: $found[ $sudoku_ref->[$i][$col]
Solver.pm|187| 39789:3:129: $found[ $sudoku_ref->[$row][$i]
Solver.pm|197| 13263:4:119: foreach my $j ( $smj .. ( $smj +
Solver.pm|141| 4445:0:20: if ( $solution_number > 0 && $solution
[Quickfix List] 2,1 Top
:~cc
  
```

# Weitere Plug-ins

## Navigation, Versions- und Projektverwaltung

- Dateibrowser: Explorer des `NETRW-Plug-ins` (Vim)
- Source Code Browser: Plug-in `taglist.vim` zusammen mit `Exuberant Ctags`
- Navigation in Perl-Bibliotheken: `Perl::Tags`
- Projektverwaltung: Plug-in `project.tar.gz`
- Versionskontrolle: Plug-in `vcsccommand.vim`
- Perl-Dokumentation: Plug-in `perl-support`
- Arbeitskontext (session) speichern/laden (Vim)
- Zuletzt geöffneten Dateien: Plug-in `mru.vim`
- ...

# Ausblick – Was fehlt?

- Integration einer voll funktionsfähigen Shell
- Integrierter Debugger
- GUI Builder (?)
- Unterstützung von Unit Tests
- Code Refactoring
- leistungsfähigere Vim-GUI

# Plugins



Fritz Mehner. *perl-support*.

<http://vim.sourceforge.net>, script 556.



Yegappan Lakshmanan. *mru.vim*.

<http://vim.sourceforge.net>, script 521.



Jeff Lanzarotta. *bufexplorer.vim*.

<http://vim.sourceforge.net>, script 42.



Aric Blumer. *project.tar.gz*.

<http://vim.sourceforge.net>, script 69.



Bob Hiestand. *vcscmd.vim*.

<http://vim.sourceforge.net>, script 90.



Yegappan Lakshmanan. *taglist.vim*.

<http://vim.sourceforge.net>, script 273.



Darren Hiebert. *Exuberant Ctags*.

<http://ctags.sourceforge.net>.

# Demo, Fragen, Kritik, Anregungen ?

---

*Vielen Dank für Ihre Aufmerksamkeit!*

