

Perl::LanguageServer & Debugger für Visual Studio Code u.a. Editoren

Gerald Richter / ECOS Technology GmbH

Deutscher Perl/Raku Workshop 2020

How it started

- - In over 20 years I use Perl, I never found a satisfying IDE for Perl
- Microsoft initiated the Language Server Protocol and Debug Adapter Protocol
- Used/usable by Visual Studio Code, Atom, Vim, Emacs,...
- Our company switched to Visual Studio Code...
- ... and Perl plugins still not very satisfying...
- ... so I wrote one that fits our needs and is hopefully usefull for others as well

Language Server

- * Syntax checking
- * Symbols in file
- * Symbols in workspace/directory
- * Goto Definition
- * Find References
- * Call Signatures
- * Supports multiple workspace folders
- * Run on remote system via ssh

Language Server

The screenshot displays the Visual Studio Code interface with a Perl Language Server. The Explorer panel on the left shows a project structure under 'BBTESTVM9C (ARBEITSBEREICH) [SSH: bbtestvm9c]'. A 'GLIEDERUNG' (Outline) view is expanded, listing symbols like 'Perl::LanguageServer', 'channel', 'debug', 'listen_port', 'log_prefix', 'log_req_txt', 'out_semaphore', 'roles', '_run_tcp_server', 'call_method', 'check_file', 'logger', 'mainloop', 'parsews', and 'process_req'. A callout bubble labeled 'Symbols in File' points to this list.

The Editor panel shows the 'LanguageServer.pm' file with Perl code. A callout bubble labeled 'Syntax Checking' points to a red squiggly line under the line `if ($reqx -> is_dapp)` at line 208. A tooltip for this line reads: 'Global symbol "\$reqx" requires explicit package name (did you forget to declare "my \$reqx ?") at - line 208. Vorschauproblem Keine Schnellkorrekturen verfügbar'.

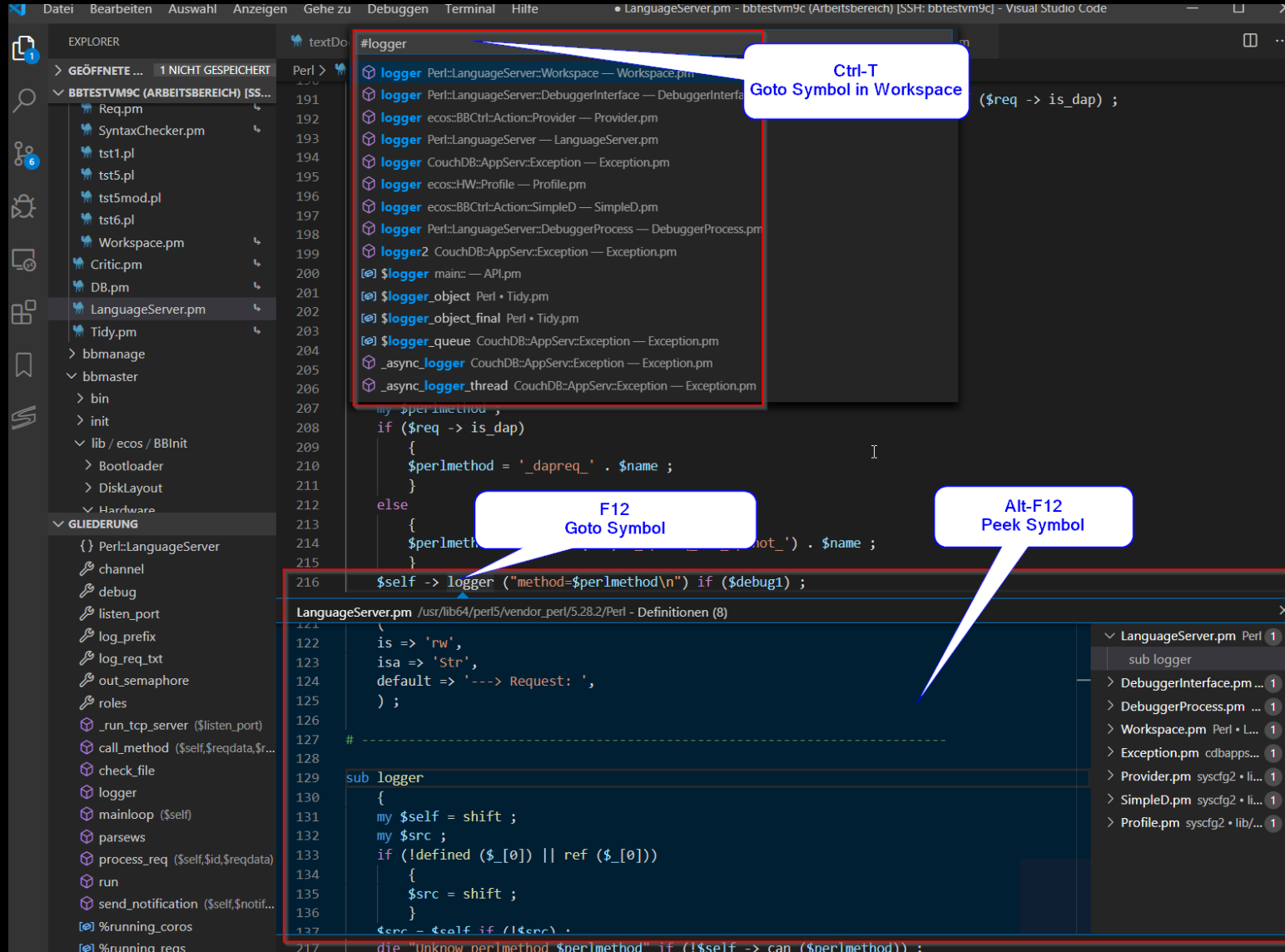
The Problems panel on the right shows two errors for 'LanguageServer.pm Perl':

- Global symbol "\$reqx" requires explicit package na... [208, 1]
- BEGIN not safe after errors--compilation aborted a... [219, 1]

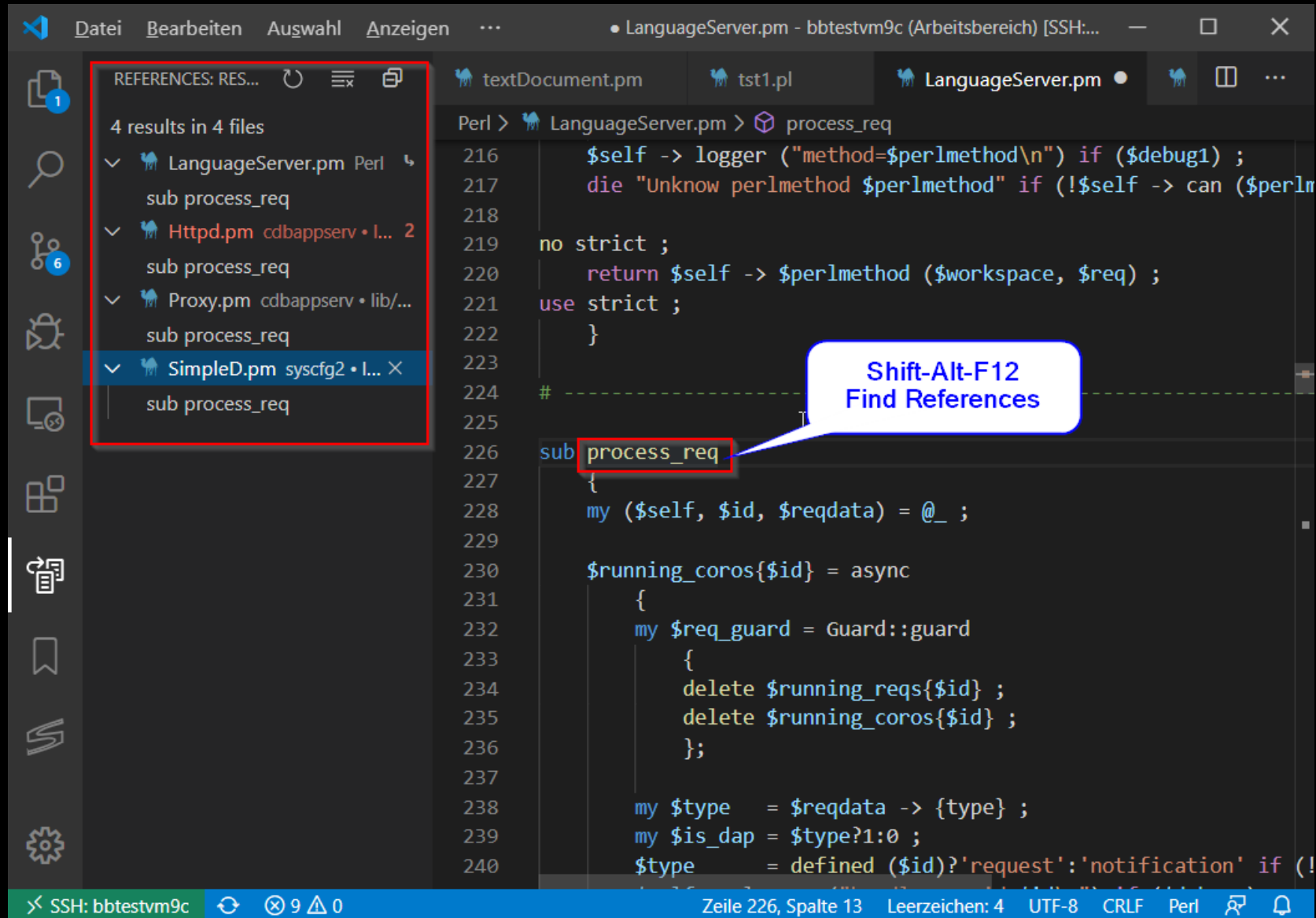
A callout bubble labeled 'Current Errors' points to this panel.

The status bar at the bottom indicates 'SSH: bbtestvm9c', '2' errors, and '0' warnings. The current cursor position is 'Zeile 205, Spalte 10'.

Language Server



Language Server



Language Server

```
# -----  
#  
# do_something - sub test  
#  
# in    $x  x value  
#      $y  y value  
# ret   nothing  
#
```

```
sub do_something  
{  
  my ($x, $y, $z) = @_ ;  
  
  print "#$x\n" ;  
  $n = -$i ;  
}
```

Parameter
Help

do_something(\$x,\$y,\$z)

do_something - sub test

Show
Comments/POD
before sub

in \$x x value
 \$y y value
ret nothing

```
do_something($a,$b
```

Call Signature Help

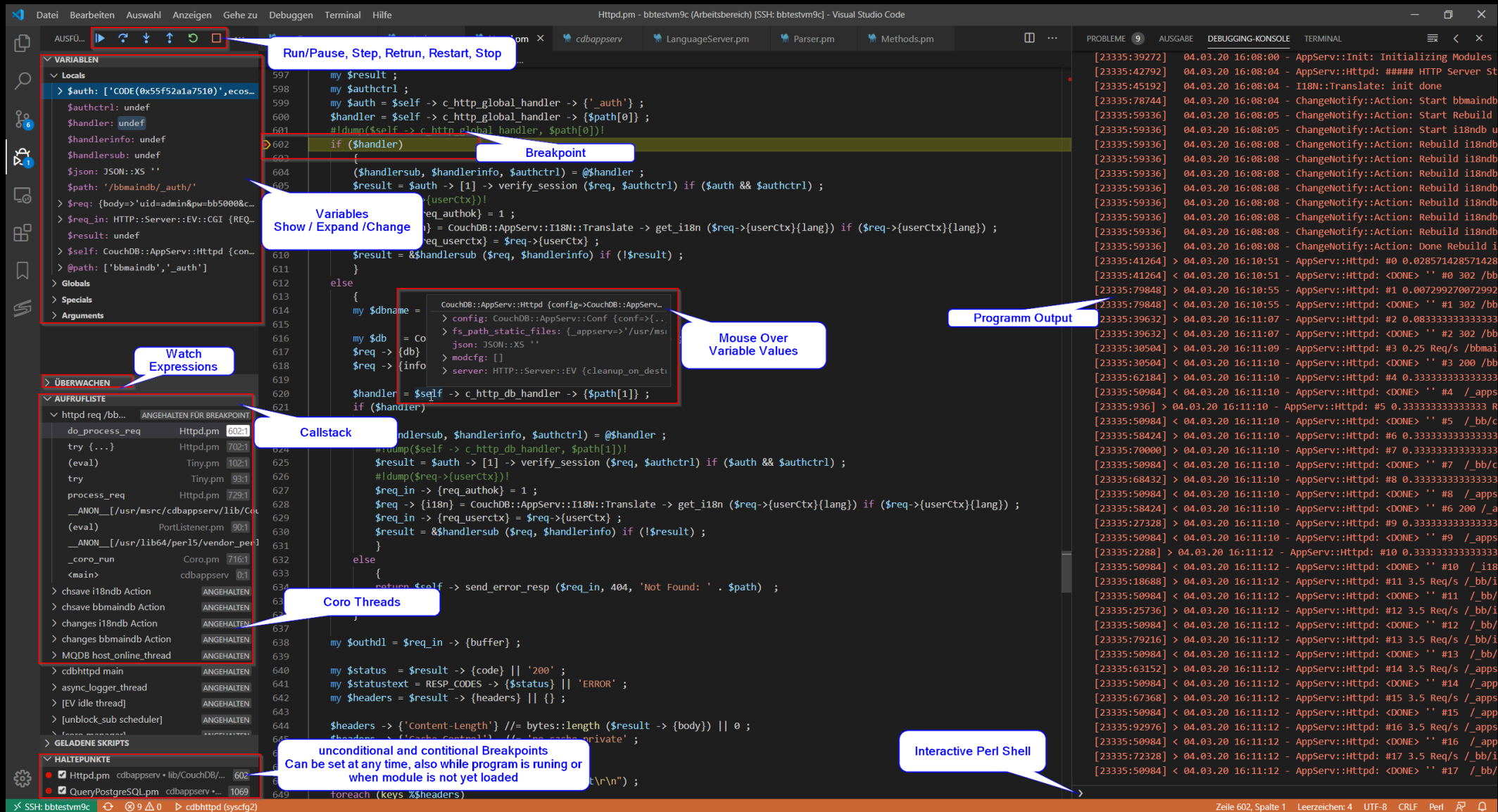
Debugger

- * Run, pause, step, next, return
- * Support for coro threads
- * Breakpoints
- * Conditional breakpoints
- * Breakpoints can be set while programm runs and for modules not yet loaded
- * Variable view, can switch to every stack frame or coro thread
- * Set variable
- * Watch variable
- * Tooltips with variable values
- * Evaluate perl code in debuggee, in context of every stack frame of coro thread
- * Automatically reload changed Perl modules while debugging
- * Debug mutiple perl programm at once
- * Run on remote system via ssh

Debugger – launch.json

```
• {  
•     "version": "0.2.0",  
•     "configurations": [  
•         {  
•             "name": "cdbhttpd",  
•             "type": "perl",  
•             "request": "launch",  
•             "program": "/usr/msrc/cdbappserv/bin/cdbappserv",  
•             "args": [ "--debug", "cdbhttpd", "--dbgflags", "secret,stacktrace" ],  
•             "reloadModules": true,  
•             "stopOnEntry": false  
•         },  
•     ]  
• }
```

Debugger



Debugger

Visual Studio Code Debugger Interface (Perl Application):

- Top Menu:** Datei, Bearbeiten, Auswahl, Anzeigen, Gehe zu, Debuggen, Terminal, Hilfe.
- Left Sidebar:** Icons for Explorer, Search, Source Control, Run and Debug, Extensions, Testing, Docker, Remote Explorer, and Settings.
- Top Bar:** cdbappserv - bbtestvm9c (Arbeitsbereich) [SSH: bbtestvm9c] - Visual Studio Code
- Central Editor:** Shows Perl code for a web server. The code includes a main function that sets up a CouchDB application and starts a web server.
- Right Sidebar:**
 - PROBLEME:** Shows a list of messages (e.g., [10033:78912] 04.03.20 17:58:59 - AppServ::Init).
 - DEBUGGING-KONSOLE:** Shows the current context and variables. The selected context is 'cdbhttpd main'.
- Bottom Panel:** Shows the 'VARIABLEN' (Variables) pane with local variables (e.g., \$basename, \$baseproc, \$block_file) and the 'AUFRUFLISTE' (Call Stack) pane showing the current call stack.

Annotations:

- Switch Context to different Coro Thread and/or Stackframe:** Points to the 'cdbhttpd main' entry in the 'AUFRUFLISTE'.
- Pause Execution at any time:** Points to the 'EV idle thread' entry in the 'AUFRUFLISTE'.
- Execute code and show/expand variables in interactive shell in current selected coro thread and stack frame:** Points to the '\$config' variable in the 'VARIABLEN' pane.

Debugger

The screenshot displays a debugger interface with three main panels:

- VARIABLEN (Variables):** Located on the left, it shows a tree structure with 'Locals' and 'Globals'. Under 'Locals', the variable `$a` is highlighted with a red circle and contains the value `1`. Other variables like `$auth`, `$authctrl`, `$handler`, `$handlerinfo`, `$handlersub`, `$json`, `$path`, `$req`, `$req_in`, `$result`, `$self`, and `@path` are listed with their respective values or types.
- Source Code:** The central panel shows the source code of `Httpd.pm`. Line 602, `my $a = 1 ;`, is highlighted with a red box. A blue callout bubble points to this line, containing the text: "Modify Code while program is running. The next time the sub is hit, module is reloaded and new code is executed. Needs: 'reloadModules': true".
- ÜBERWACHEN (Watch) and AUFRUFLISTE (Call Stack):** The bottom panel shows the call stack. The current frame is `do_process_req` at line 603:1 in `Httpd.pm`, which is marked as 'ANGEHALTEN FÜR BREAKPOINT' (Stopped at breakpoint).

The source code in the center panel includes various Perl functions and logic, such as `$req -> {body_decoded} = $json->decode($req_in -> {REQUEST_BODY}) ;`, `my $auth = $self -> c_http_global_handler -> {'_auth'} ;`, and `my $handler = $self -> c_http_global_handler -> {$path[0]} ;`.

Installation

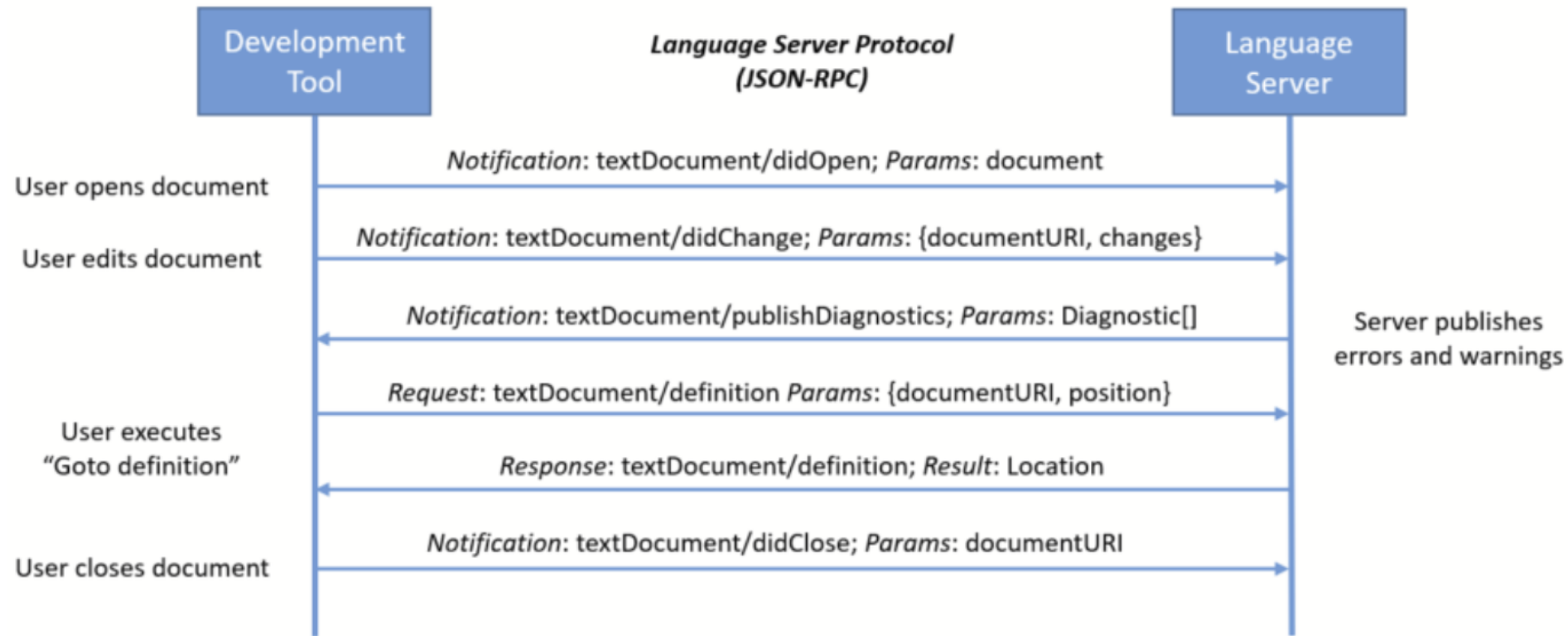
- Consists of two Parts:
 - Plugin for IDE
 - Language Server written in Perl
- Visual Studio Code Plugin „perl“
- Perl::LanguageServer
 - Compiler::Lexer is now updated on CPAN
- Set Debug Adapter Port for multiple debug sessions

Remote via ssh

- `"sshAddr": "10.11.12.13",`
- `"sshUser": "root"`
- Example: if your local path is `\\10.11.12.13\share\path\to\ws` and on the remote machine you have `/path/to/ws`
- `"sshWorkspaceRoot": "/path/to/ws"`
- Or use `pathMap`
- `"perl.pathMap": [`
- `[`
- `"file:///",`
- `"file:///home/systems/mountpoint/"`
- `]`
- `]`

How the Language Server works

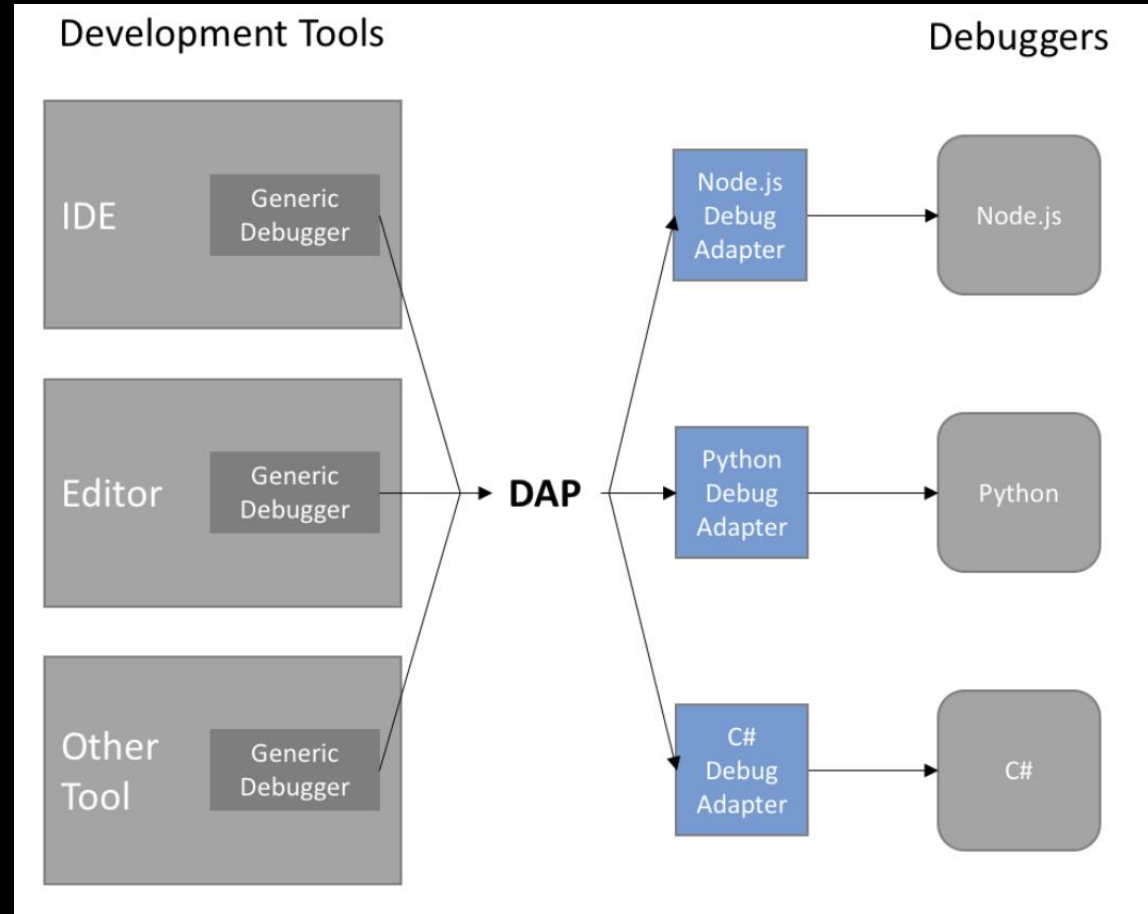
A language server runs as a separate process and development tools communicate with the server using the language protocol over JSON-RPC. Below is an example for how a tool and a language server communicate during a routine editing session:



How to extent Language Server

- Every Request in the protocol coresponds to a method in `Perl::LanguageServer::Methods`
- So `textDocument/signatureHelp` calls `Perl::LanguageServer::Methods::textDocument::_rpcreq_signatureHelp`

How Debug Adapter Protocol works



<https://microsoft.github.io/debug-adapter-protocol/overview>

How the debugger is implemented

- Stub in IDE communicates via „Debug Adapter Port“ with `Perl::LanguageServer::Methods::DebugAdapter` which is part of the Language Server
- `Perl::LanguageServer::Methods::DebugAdapterInterface` starts and communicates with Debuggee
- Debuggee runs under control of `Perl::LanguageServer::DebuggerInterface`



Using other Editors

- Clients for Debug Adapter Protocol and Language Server Protocol are available for other IDEs as well
- Needs to be documented

You are welcome to contribute...