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

Gerald Richter / ECOS Technology GmbH

Deutscher Perl/Raku Workshop 2020

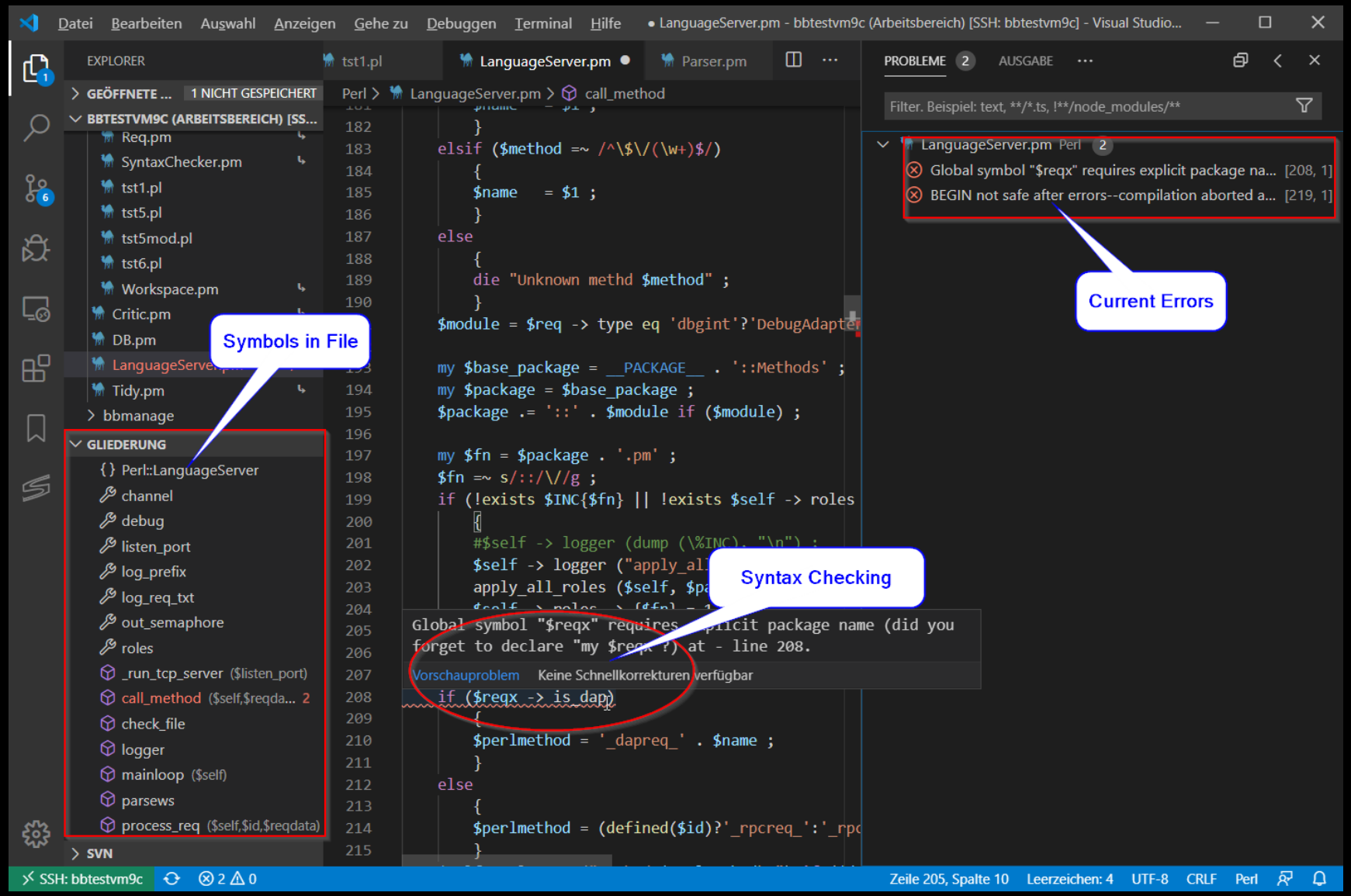
How it started

- 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

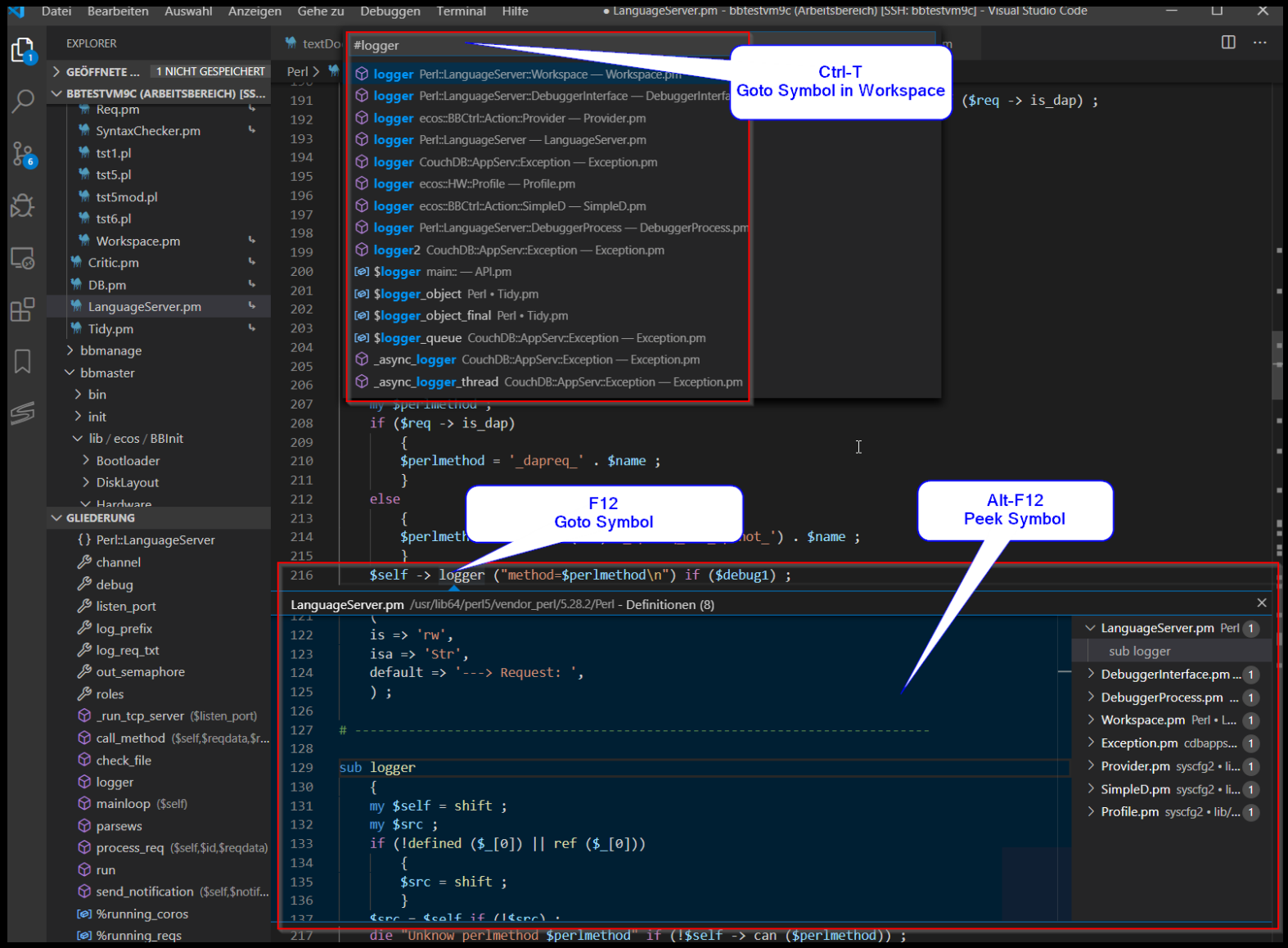
Language Server

- * 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

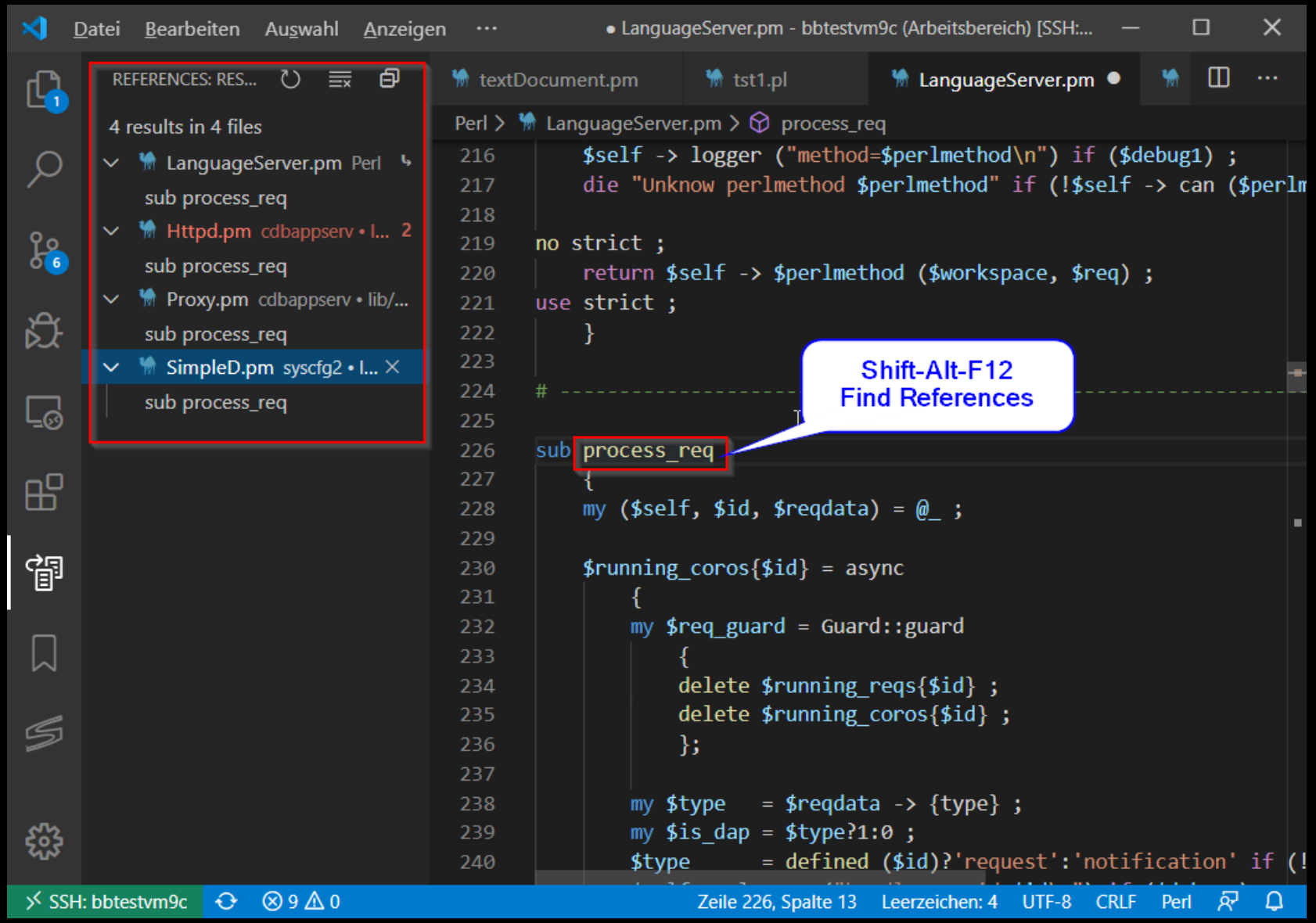
Language Server



Language Server

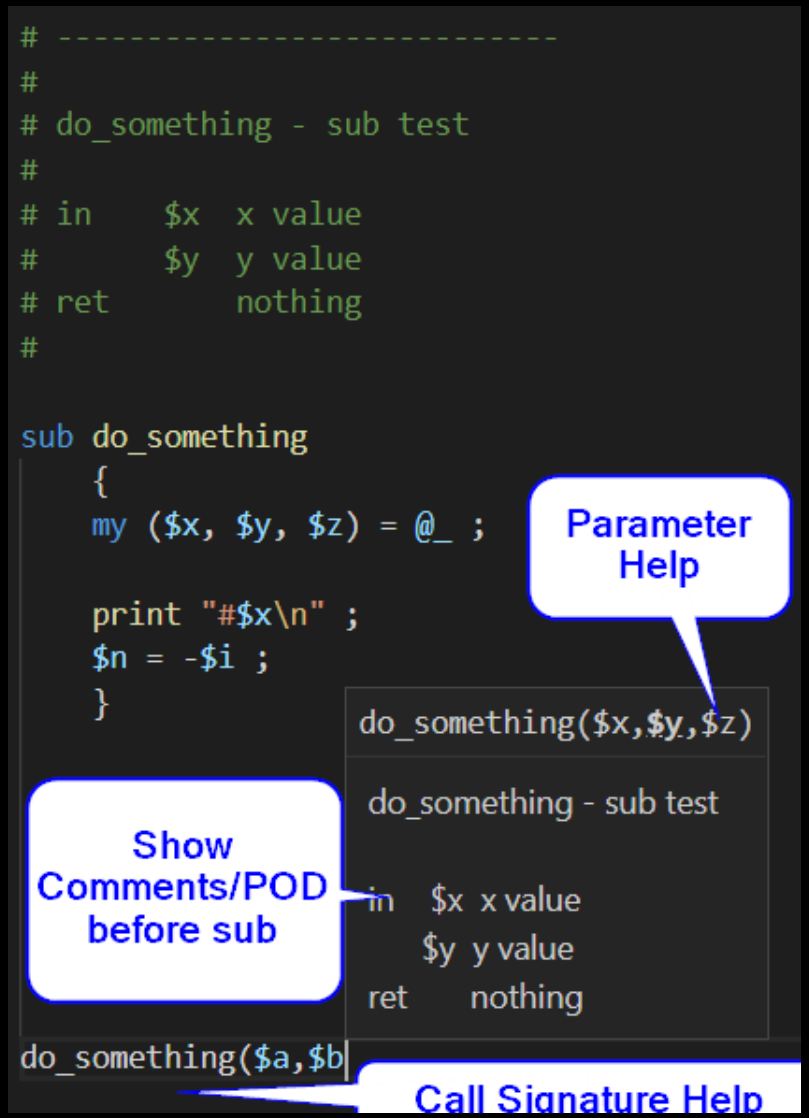


Language Server




```
216     $self -> logger ("method=$perlmethod\n") if ($debug1) ;
217     die "Unknow perlmethod $perlmethod" if (!$self -> can ($perlm
218
219 no strict ;
220     return $self -> $perlmethod ($workspace, $req) ;
221 use strict ;
222     }
223
224 # ----------
225
226 sub process_req
227     {
228     my ($self, $id, $reqdata) = @_ ;
229
230     $running_coros{$id} = async
231         {
232         my $req_guard = Guard::guard
233             {
234             delete $running_reqs{$id} ;
235             delete $running_coros{$id} ;
236             };
237
238         my $type    = $reqdata -> {type} ;
239         my $is_dap = $type?1:0 ;
240         $type      = defined ($id)?'request':'notification' if (!
```

Language Server



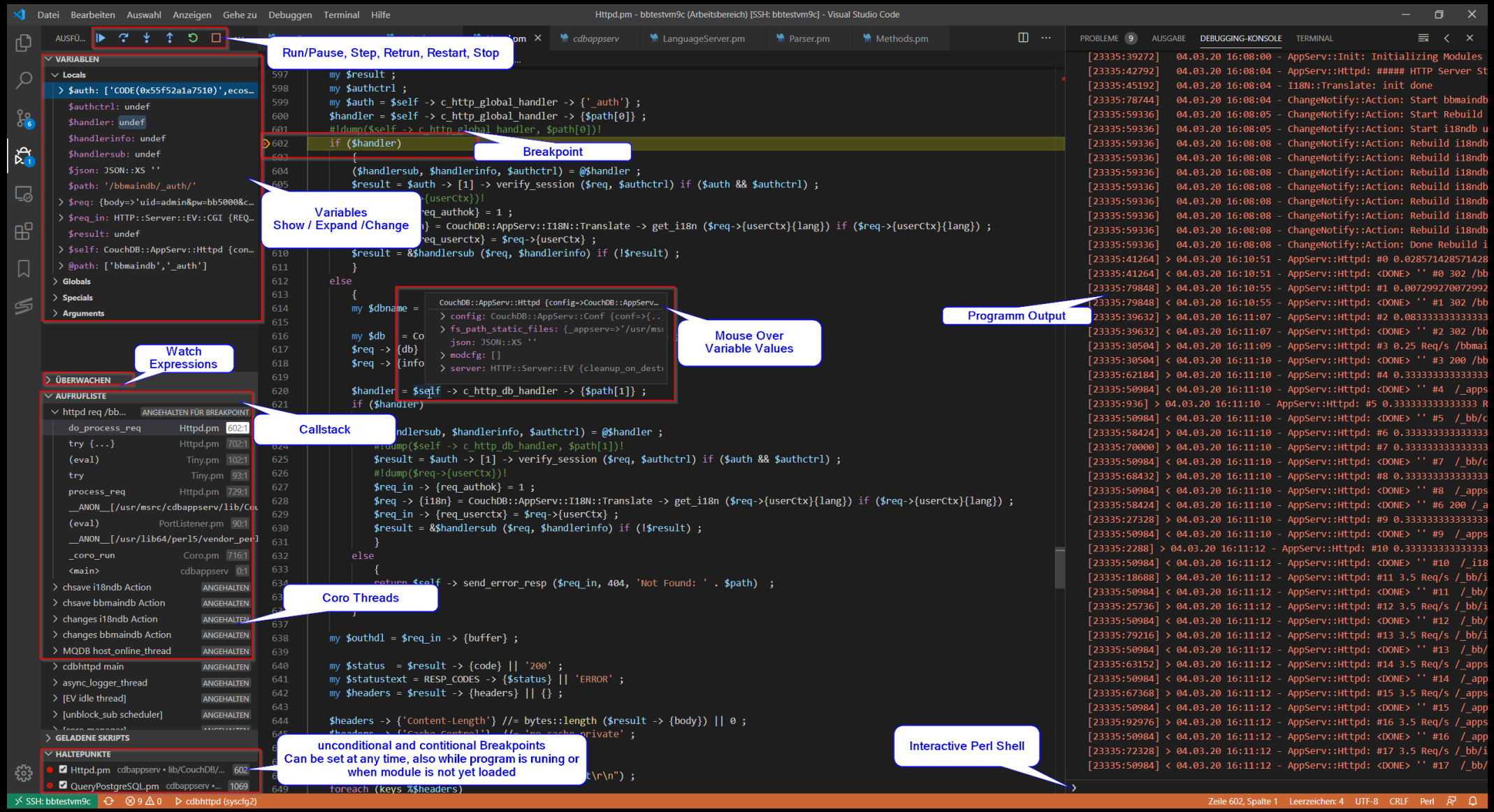
Debugger

- * 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

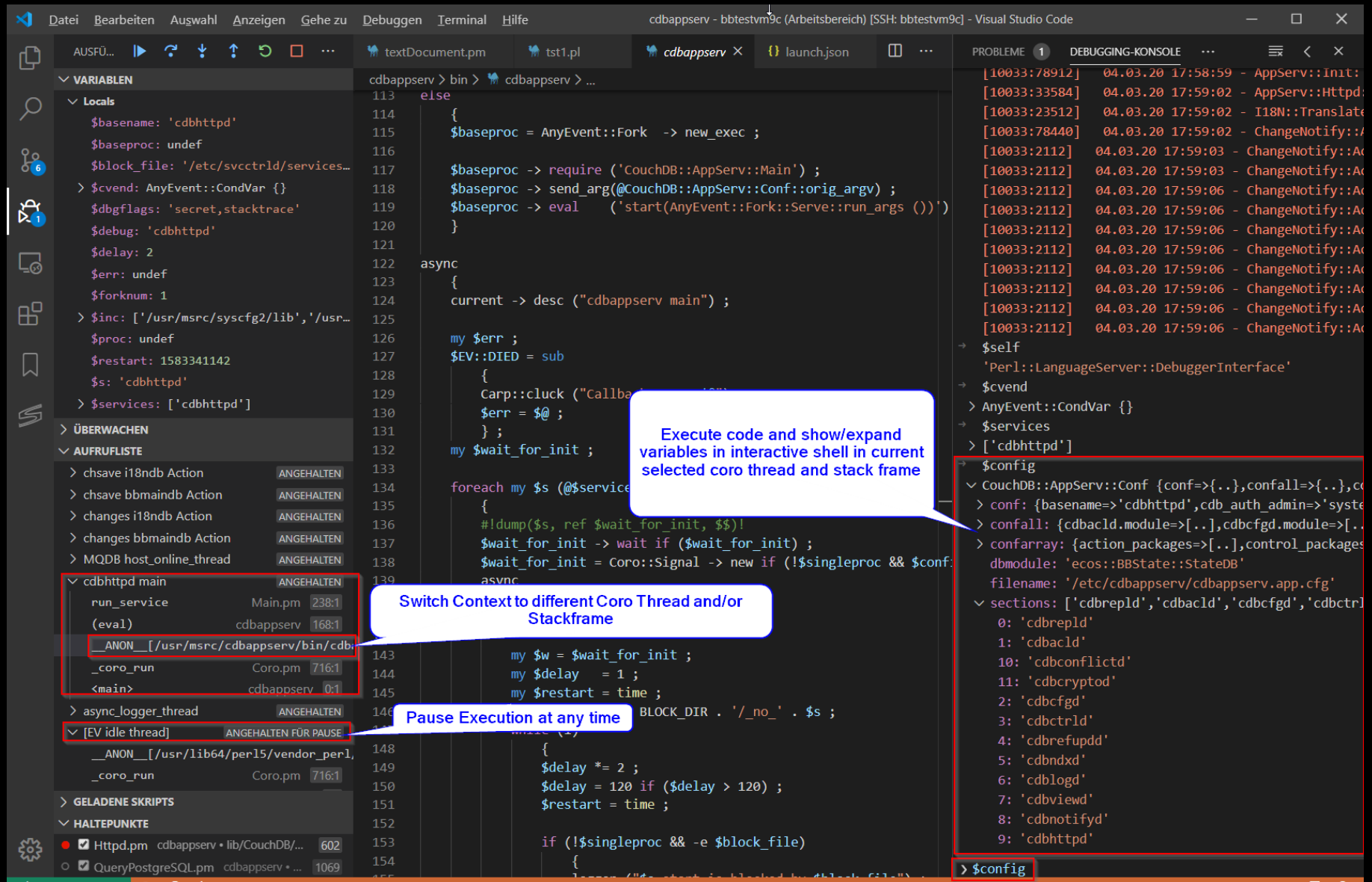
Debugger – launch.json

```
{
    "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
        },
    ]
}
```

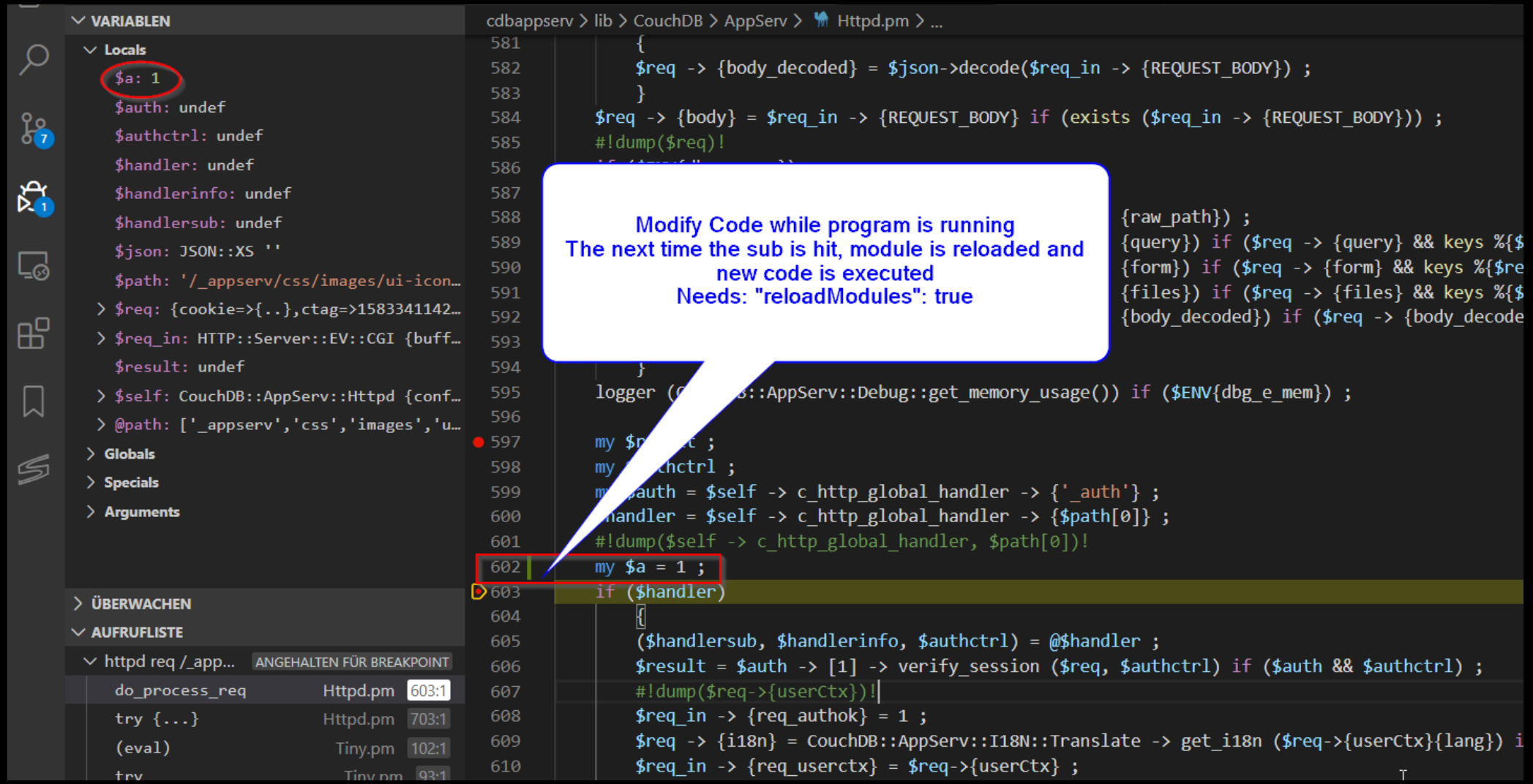
Debugger


Run/Pause, Step, Retrun, Restart, Stop
Breakpoint
Variables
Show / Expand /Change
Watch
Expressions
Callstack
Coro Threads
Mouse Over
Variable Values
Programm Output
unconditional and contitional Breakpoints
Can be set at any time, also while program is runing or
when module is not yet loaded
Interactive Perl Shell

Debugger


Execute code and show/expand variables in interactive shell in current selected coro thread and stack frame
Switch Context to different Coro Thread and/or Stackframe
Pause Execution at any time

Debugger


VARIABLEN
Locals
$a: 1
$auth: undef
$authctrl: undef
$handler: undef
$handlerinfo: undef
$handlersub: undef
$json: JSON::XS ''
$path: '/_appserv/css/images/ui-icon…
$req: {cookie=>{..},ctag=>1583341142…
$req_in: HTTP::Server::EV::CGI {buff…
$result: undef
$self: CouchDB::AppServ::Httpd {conf…
@path: ['_appserv','css','images','u…
Globals
Specials
Arguments
ÜBERWACHEN
AUFRUFLISTE
httpd req /_app... ANGEHALTEN FÜR BREAKPOINT
do_process_req Httpd.pm 603:1
try {...} Httpd.pm 703:1
(eval) Tiny.pm 102:1
try Tiny.pm 93:1
cdbappserv > lib > CouchDB > AppServ > Httpd.pm > ...
Modify Code while program is running
The next time the sub is hit, module is reloaded and new code is executed
Needs: "reloadModules": true
my $a = 1 ;
if ($handler)
($handlersub, $handlerinfo, $authctrl) = @$handler ;
$result = $auth -> [1] -> verify_session ($req, $authctrl) if ($auth && $authctrl) ;
#!dump($req->{userCtx})!
$req_in -> {req_authok} = 1 ;
$req -> {i18n} = CouchDB::AppServ::I18N::Translate -> get_i18n ($req->{userCtx}{lang}) i
$req_in -> {req_userctx} = $req->{userCtx} ;

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

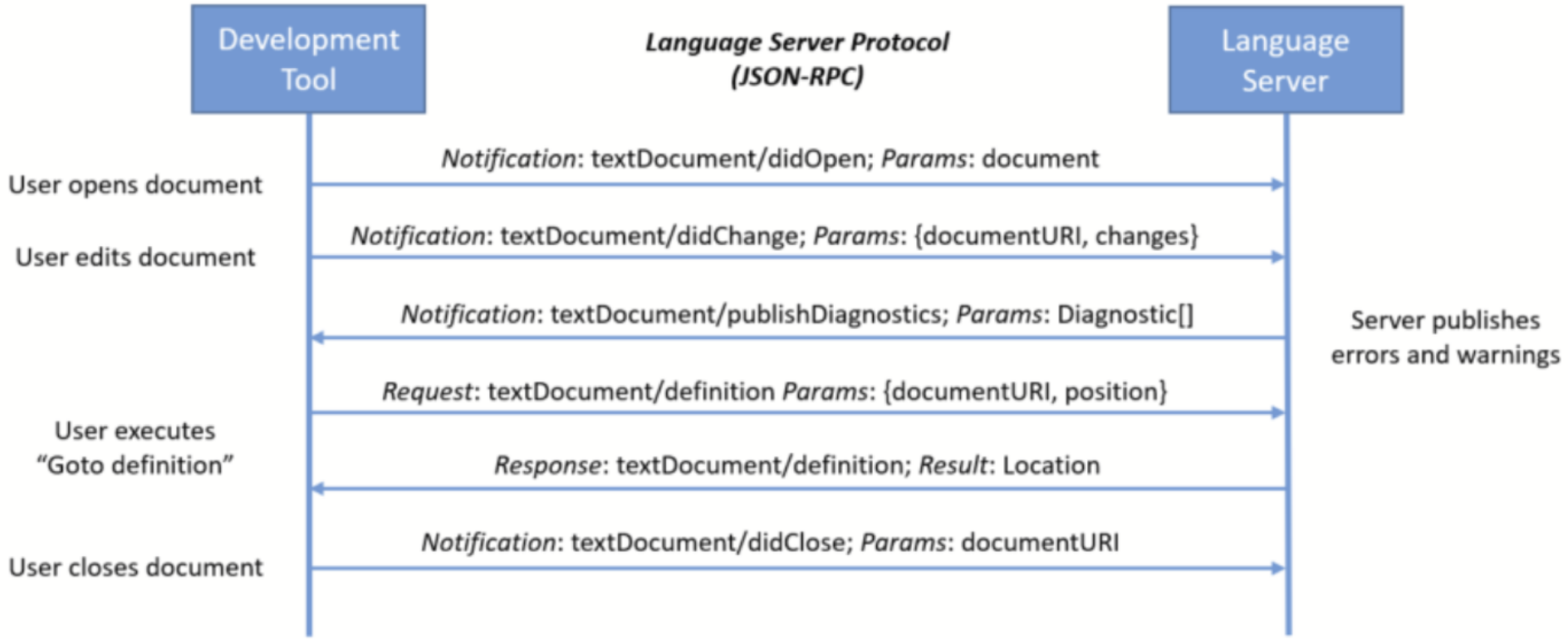
- Set Debug Adapter Port for multiple debug sessions

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:

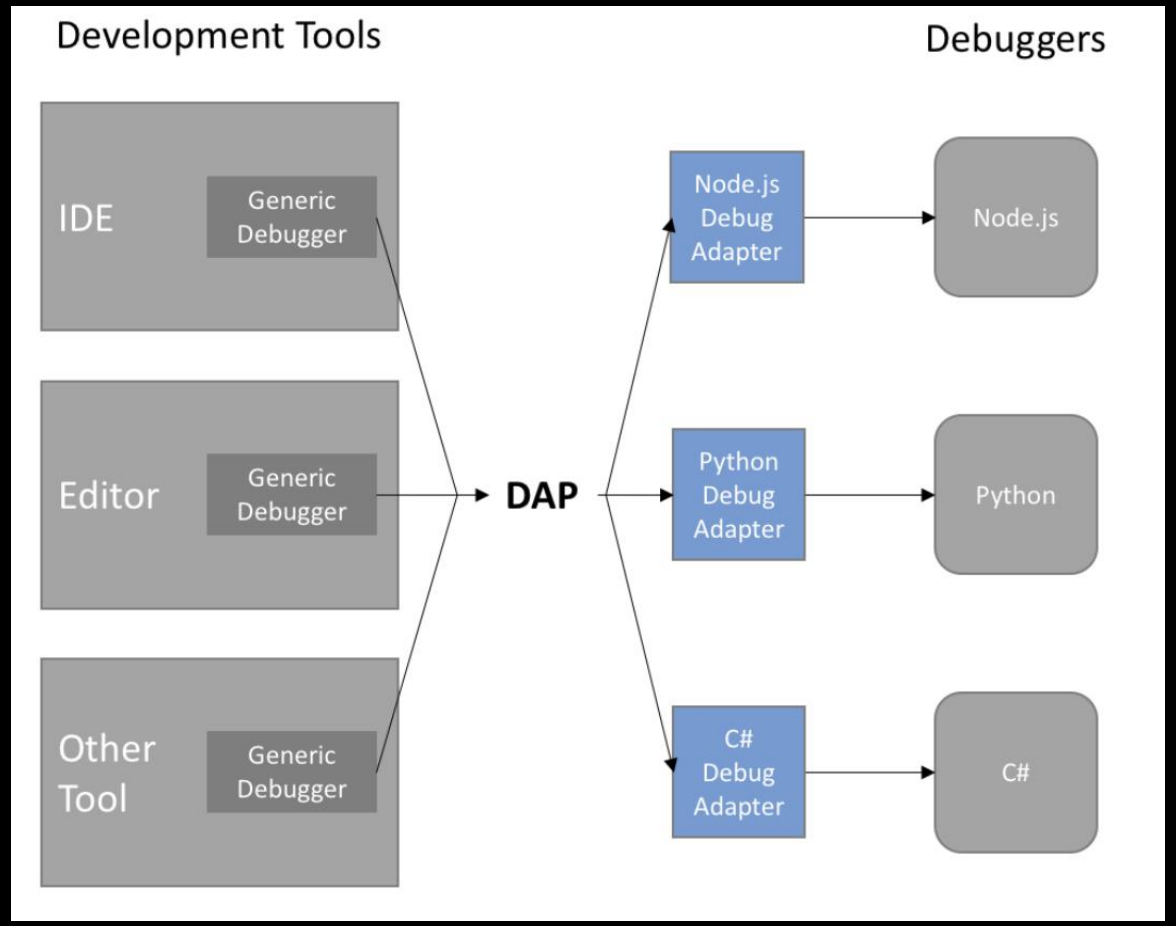


https://microsoft.github.io/language-server-protocol/overviews/lsp/overview

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

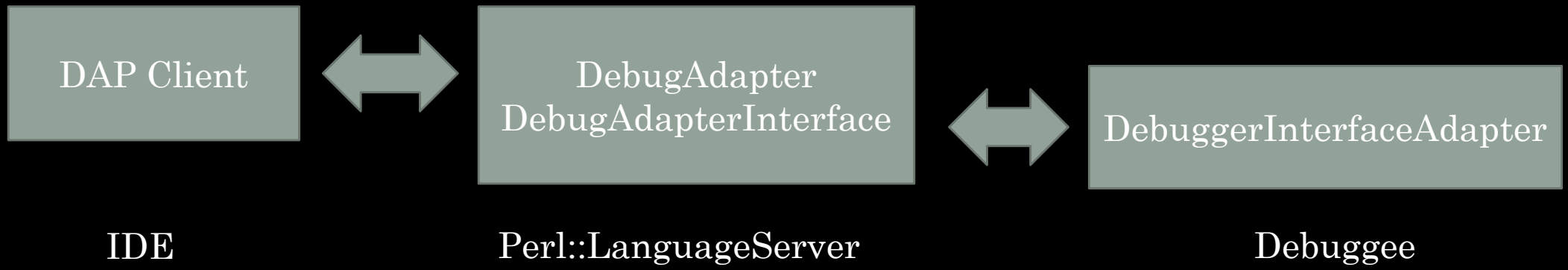
How Debug Adapter Protocol works



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

How the debugger is implemented

- Stub in IDE communicates via „Debug Adapater 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 documentated

You are welcome to contribute…