

Když něco rozeberem, tak leda debuggerem



WebExpo 2009 – Praha

Mgr. Juraj Michálek
SinusGear



Twitter: <http://twitter.com/georgiksk>

Blog: <http://georgik.sinusgear.com>

Tak čo rozoberieme?

Bug

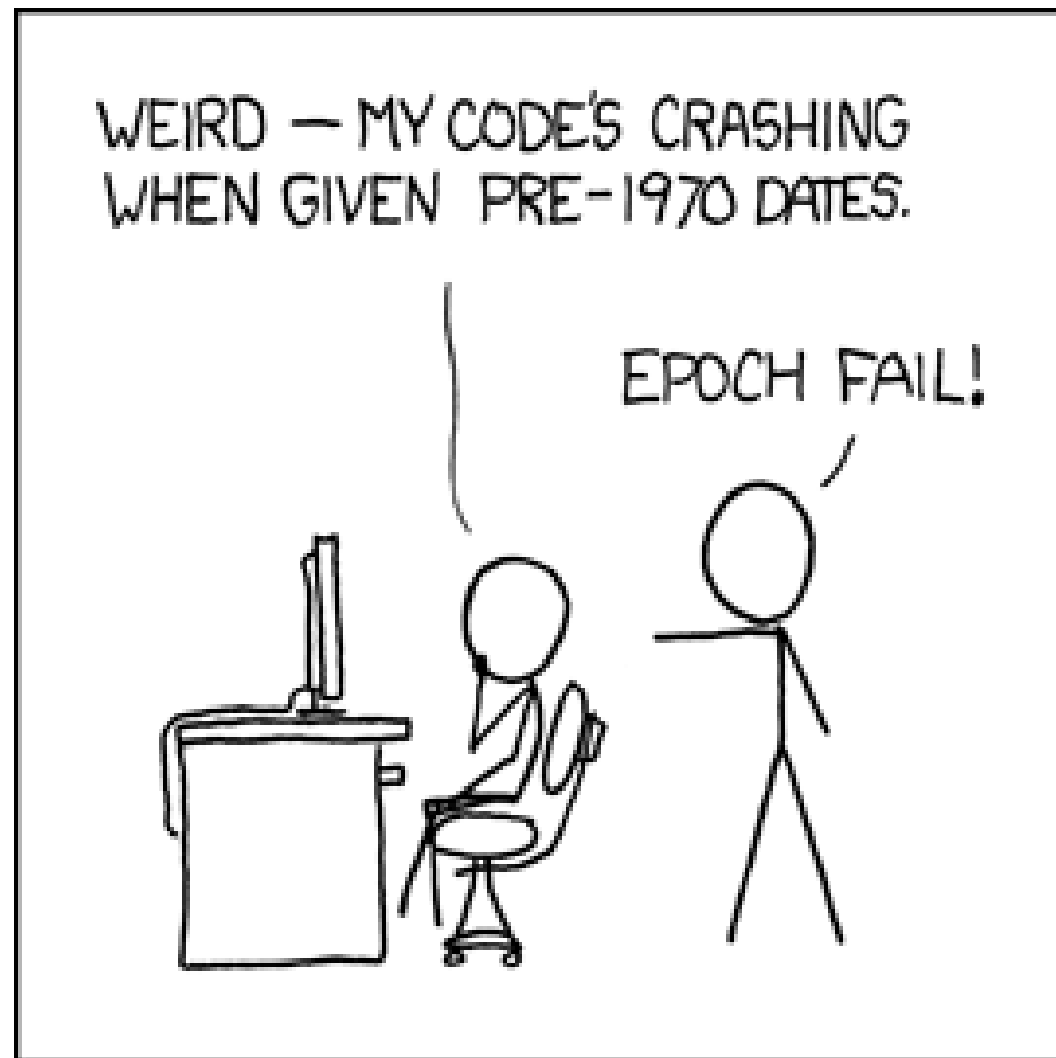
Ladenie

Debuggovanie
(lokálne aj vzdialené)

Bez debuggeru a predsa bezpečne

Zaujímavosti

Bug



Čím trávi vývojár najviac času?

Ladením.

Debuggovaním.

Facebookom ;-)

Ladíme

Prvý přístup:

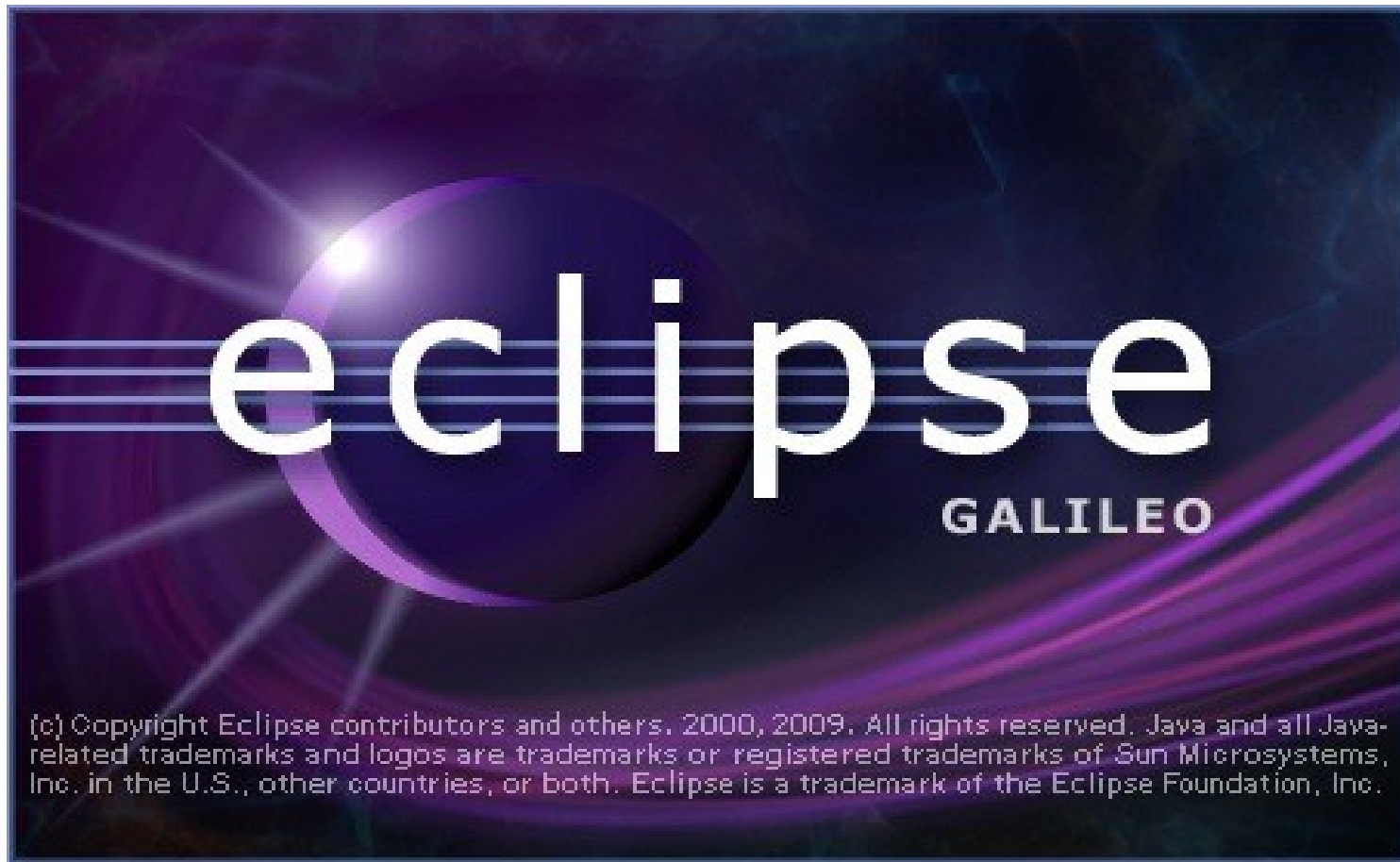
Logujeme.

Debugger



Krok a stop.
Krok a stop.
Krok a stop.

Uvedené příklady z Eclipse



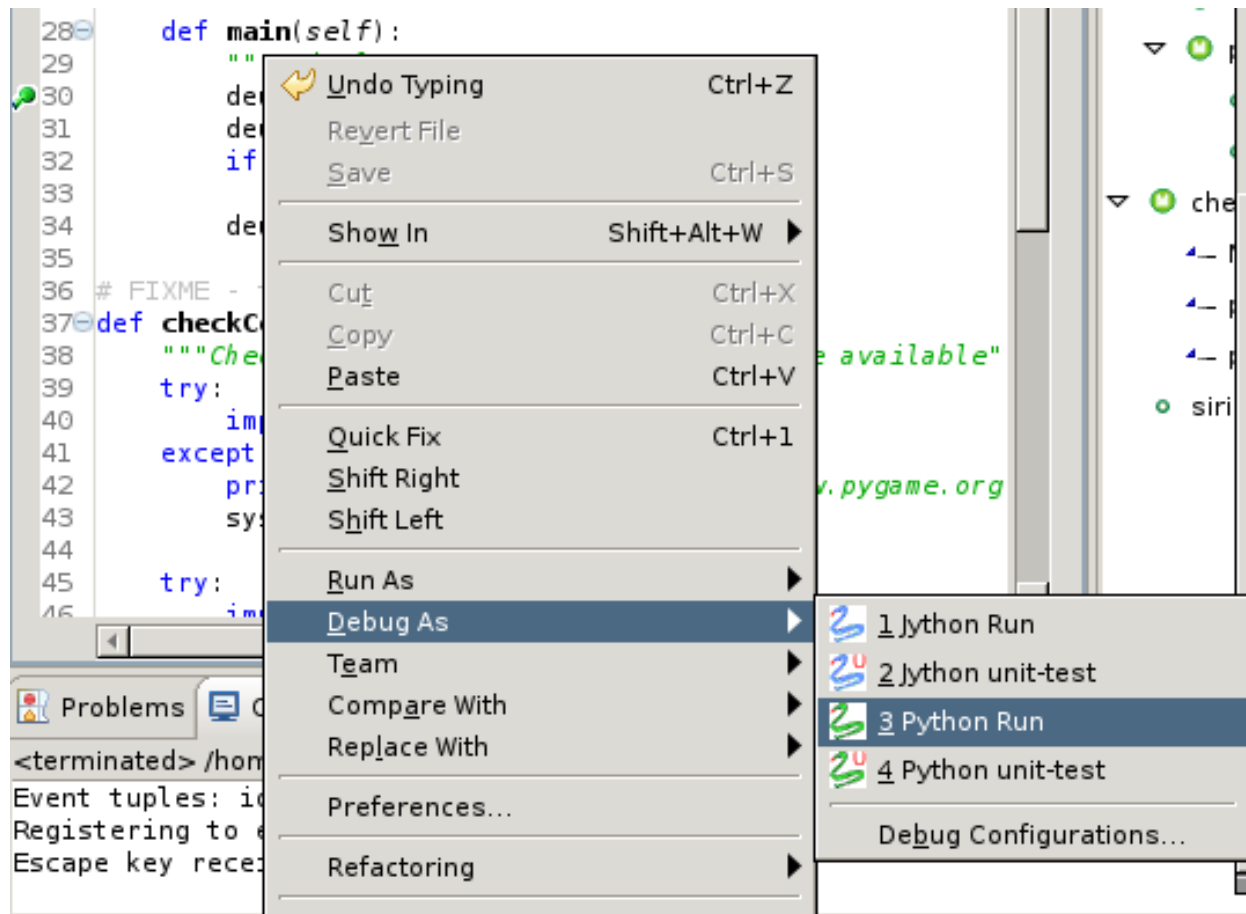
Break point



```
27  
28  
29  
30  
31  
32  
33  
34  
35
```

```
def main(self):  
    """Main loop"""  
    deus = ExMachina()  
    deus.init()  
    if not self.quickQuit:  
        deus.run(self.commandMode)  
    deus.done()
```


Naštartujeme Debugger



A zastavíme

The screenshot displays a Python IDE in a debug state. The 'Variables' window is open, showing the following data:

Name	Value
sys	module. <mod...
• __egginsert	int: 60
• __plen	int: 13
• api_version	int: 1013
• argv	list: ['/home/g...
• 0	str: /home/ge...

The code editor shows the following Python code:

```
26         print "Quick quit mode"
27
28     def main(self):
29         """Main loop"""
30         deus = ExMachina()
31         deus.init()
32         if not self.quickQuit:
33             deus.run(self.consoleMode)
```

Prechádzka po stacku

The screenshot displays a Python IDE interface during a debug session. The top-left pane shows the 'Debug' window with a tree view of the execution stack. The top-right pane shows the 'Variables' window with a table of current variables. The bottom pane shows the source code of the file 'Siriel.py'.

Debug Window (Stack Trace):

- siriel5 Siriel.py [Python Run]
- Siriel.py
 - pydevd.reader
 - pydevd.writer
 - MainThread
 - main [Siriel.py:30]
 - <module> [Siriel.py:66]
 - run [pydevd.py:689]
 - <module> [pydevd.py:852]

Variables Window:

Name	Value
ExMachina	ExMachina.Ext
type	str: classobj
Siriel	__main__.Siriel
type	str: classobj
__builtins__	dict: {'IndexEr
doc	NoneType: No

Source Code (Siriel.py):

```
64 siriel = Siriel()
65 siriel.processArg()
66 siriel.main()
67
68
69 # vim: expandtab:smarttab:shiftwidth=4
70
```

Výhody dobrého debuggera

Zastavenie v mieste break-pointu.

Vizualizácia hodnôt.

Stack trace.

Ladíme lokálnu aplikáciu

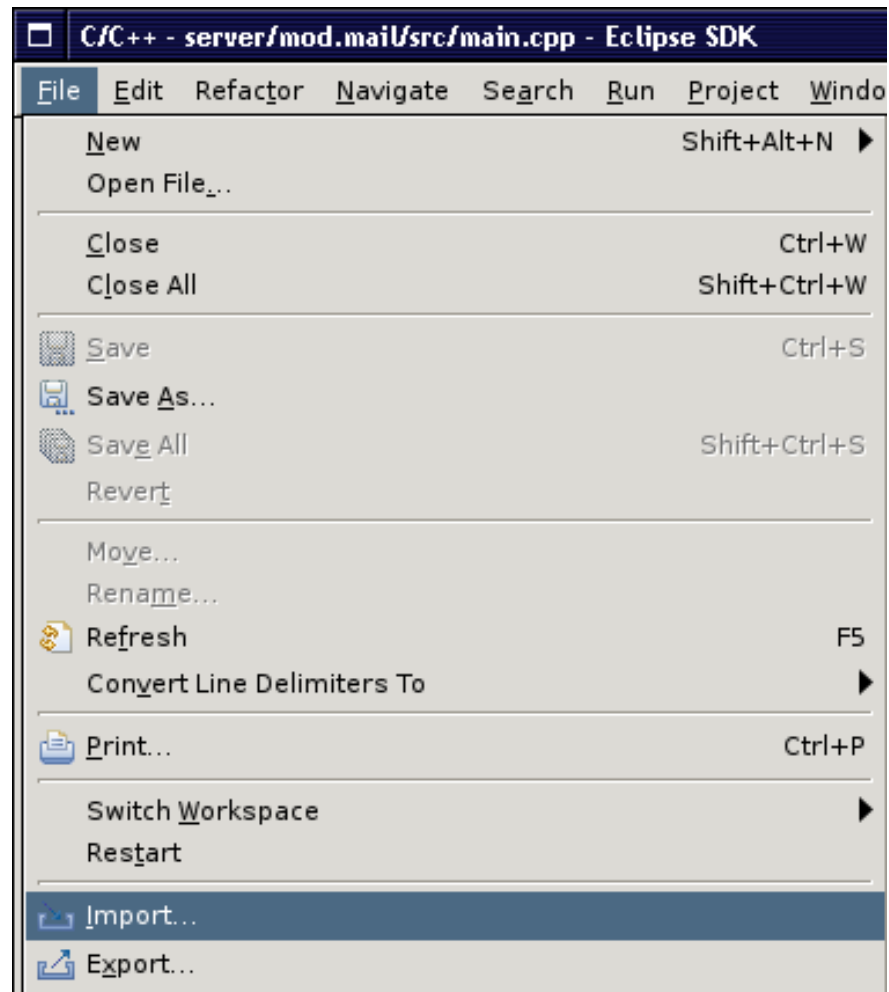
Možnosti:

Aplikácia vrámci workspace.
(videli sme)

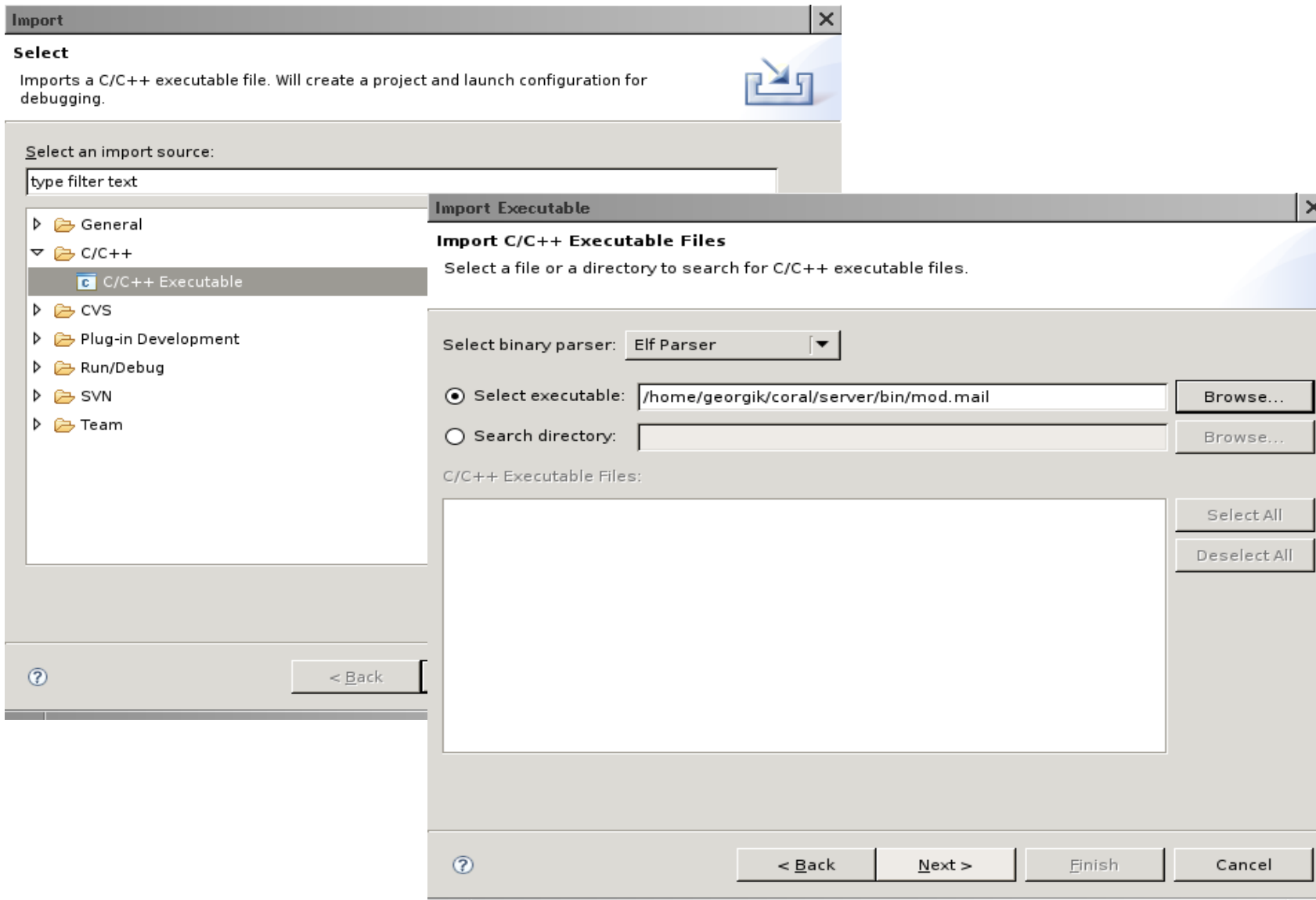
Aplikácia mimo workspace.
(ukážeme si)



Import binárky



Toto ešte zvládneme



A teraz príde trik – bez medzier!

New project name:
The new project will let you debug but not build the executable.

Existing project:

Create a Launch Configuration: ▾

Name:

Nesprávne!

Správne

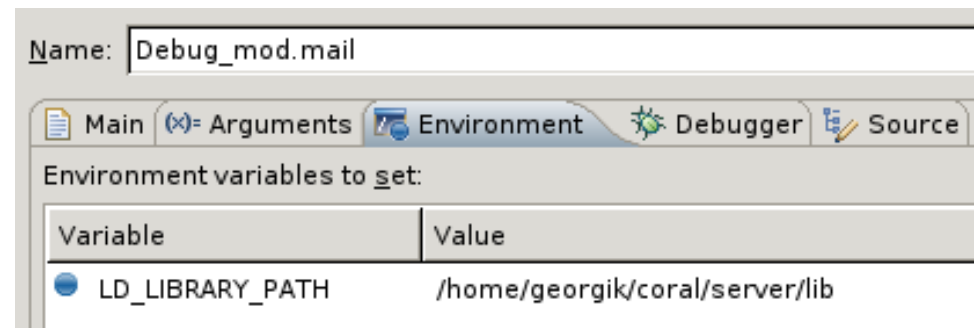
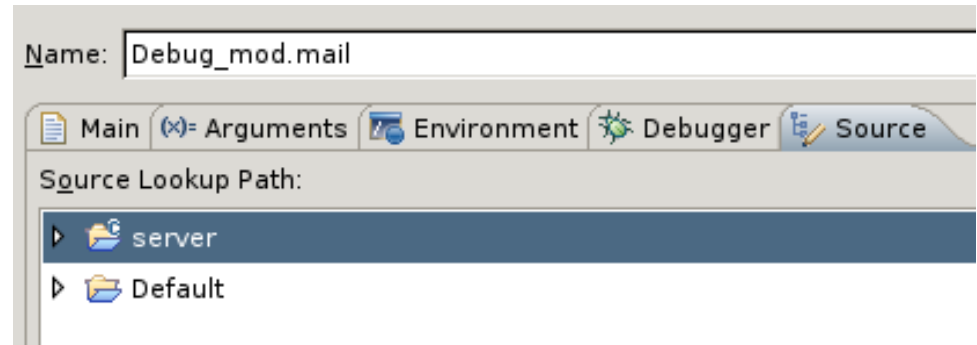
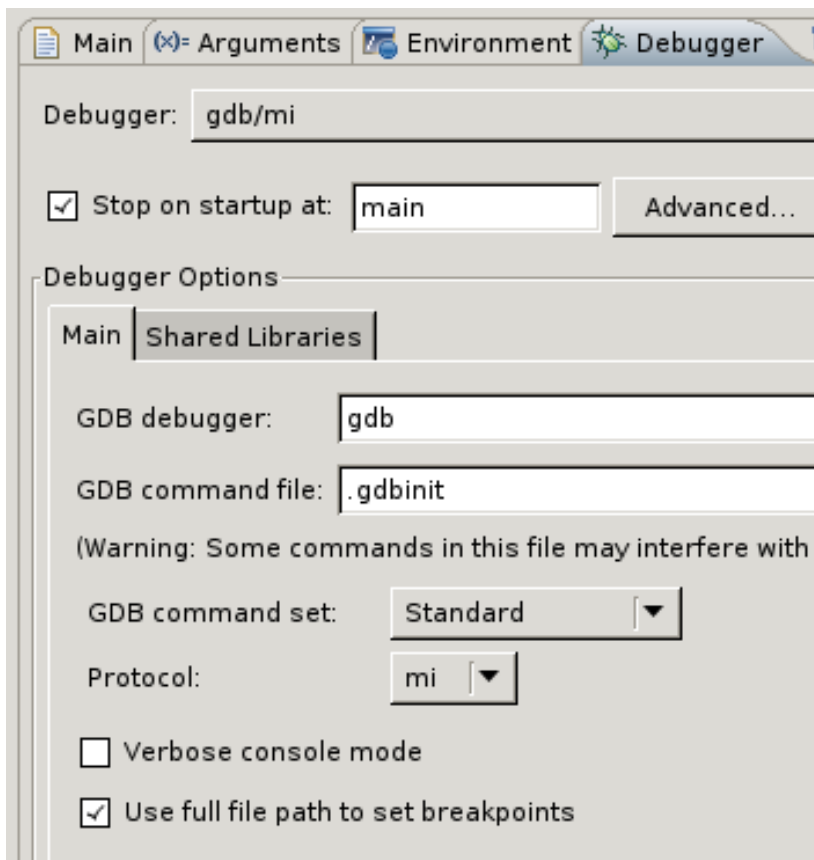
New project name:
The new project will let you debug but not build the executable.

Existing project:

Create a Launch Configuration: ▾

Name:

Drobnosti pred štartom



Debug – lokálna aplikácia

The screenshot shows a debugger interface with the following components:

- Debug Window:** Shows the execution stack. The current thread is suspended at line 2 of `ParseArgv()` in `/home/georgik/coral/server`. Below it, line 1 of `main()` in `/home/georgik/coral/server/dblay` is visible. The debugger is identified as `gdb (10/15/09 8:09 AM)`.
- Variables Window:** Lists the current state of variables:
 - `argc`: integer value
 - `argv`: array of strings
 - `a`: pointer to `ARGVParsed`
 - `module`: string value
- Source Code Window:** Displays the code for `main.cpp`. A breakpoint is set at line 20:

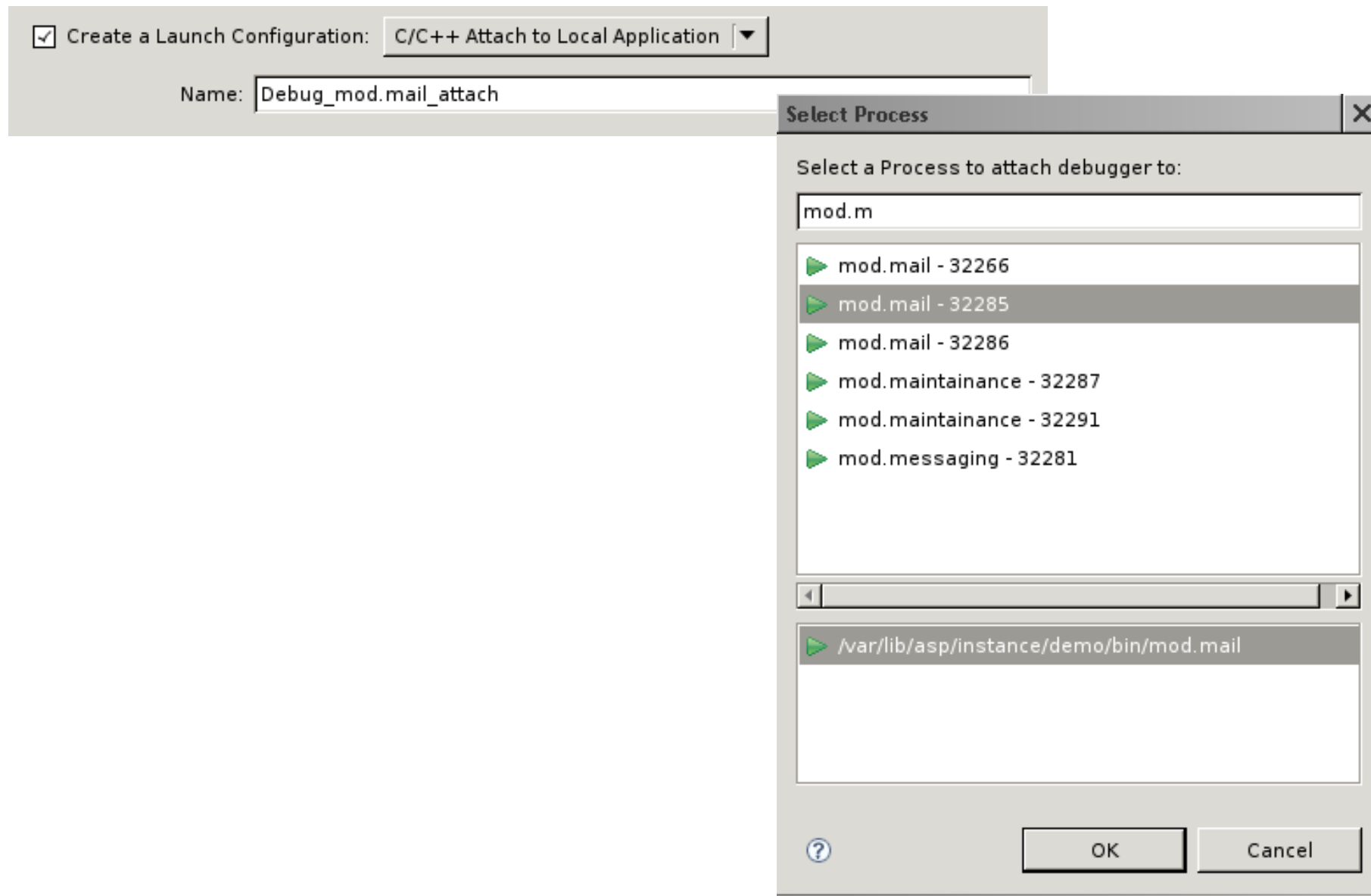
```
17 void ParseArgv(int argc, char *argv[], ARGVParsed *a, const char *module)
18 {
19     int i;
20     memset(a, 0, sizeof(ARGVParsed));
21     for(i=1; i<argc; i++)
22     {
23         if(xstrcmp("-?", argv[i])==0 || xstrcmp("-help", argv[i])==0 || xstrcmp("--help", arg
24         {
```

Čo z bežiacimi aplikáciami?



Pripojíme sa na aplikáciu.

Pripojenie k bežiacemu procesu



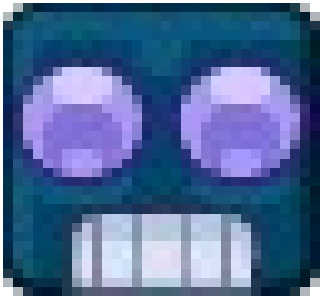
Čo s aplikáciami na serveri?



Väčšina debuggerov podporuje vzdialené pripojenie.



Aplikácia to neprežila



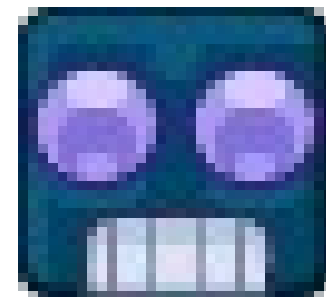
Post-mortem analýza

Zostal len jej obraz: coredump

Ukladanie pádov Apache

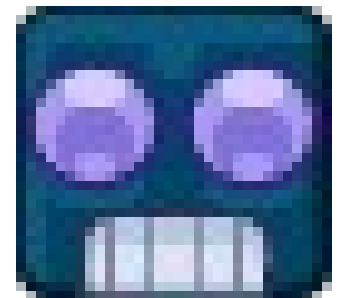
Do konfigurácie pridať:

`CoreDumpDirectory /tmp/`



Rýchla analýza coredumpu

```
(georgik@traper:pts/6)~--(/tmp)
(10:43: )~--(Št,okt15)
(georgik@traper:pts/6)~--(/tmp)
(10:44: )~--(Št,okt15)
[1] 532
(georgik@traper:pts/6)~--(/tmp)
(10:44: )~--(Št,okt15)
[1] + quit (core dumped) xeyes
(georgik@traper:pts/6)~--(/tmp)
(10:44: )~--(Št,okt15)
(gdb) gdb /usr/bin/xeyes core
(no debugging symbols found)
Core was generated by `xeyes'.
Program terminated with signal 3, Quit.
#0  0xb802f424 in __kernel_vsyscall ()
(gdb) bt full ←
#0  0xb802f424 in __kernel_vsyscall ()
No symbol table info available.
#1  0xb7dbd42b in poll () from /lib/i686/cmov/libc.so.6
No symbol table info available.
#2  0xb7fdb2f7 in _XtWaitForSomething () from /usr/lib/libXt.so.6
No symbol table info available.
#3  0xb7fdc75b in XtAppNextEvent () from /usr/lib/libXt.so.6
```



Módy debuggera

Spustenie lokálnej aplikácie.

Pripojenie k lokálnemu procesu.

Pripojenie k vzdialenému procesu.

Post-mortem.

Debugujeme PHP



Eclipse + PHP Development Tools

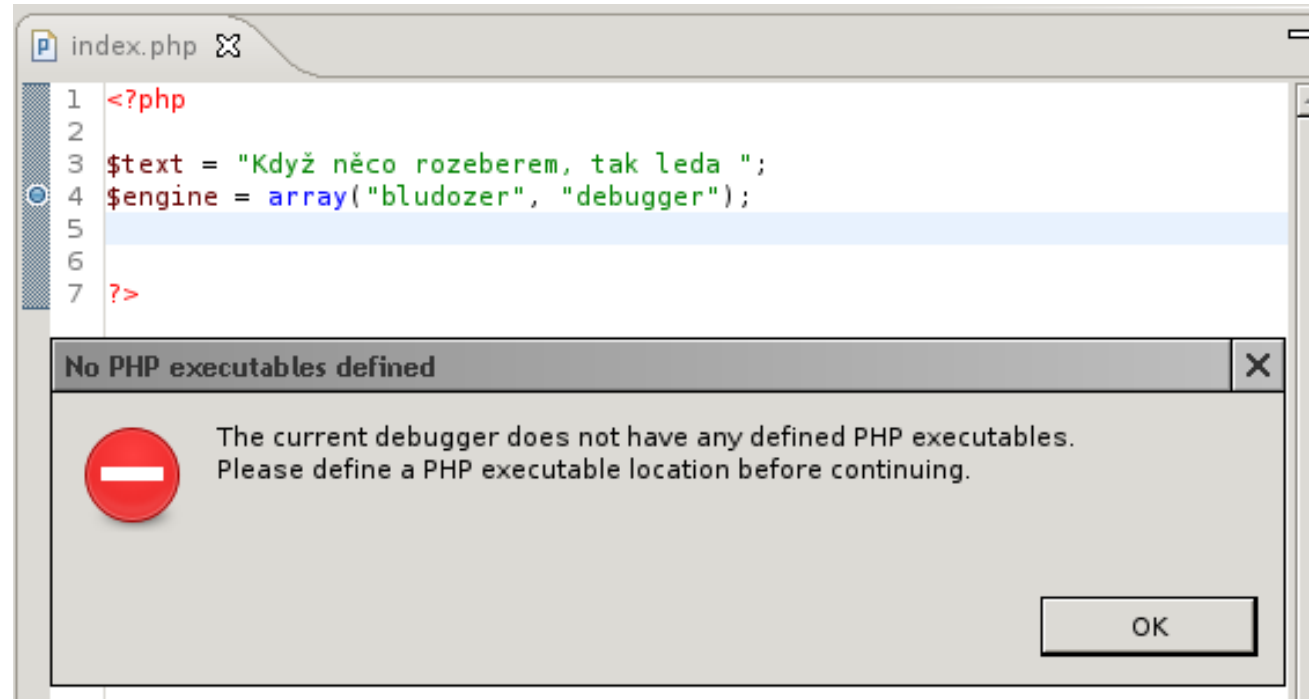


Eclipse for PHP Developers (138 MB)

Tools for PHP developers creating Web applications, including PHP Development Tools (PDT), Web Tools Platform, Mylyn and others. [More...](#)

Downloads: 80,849

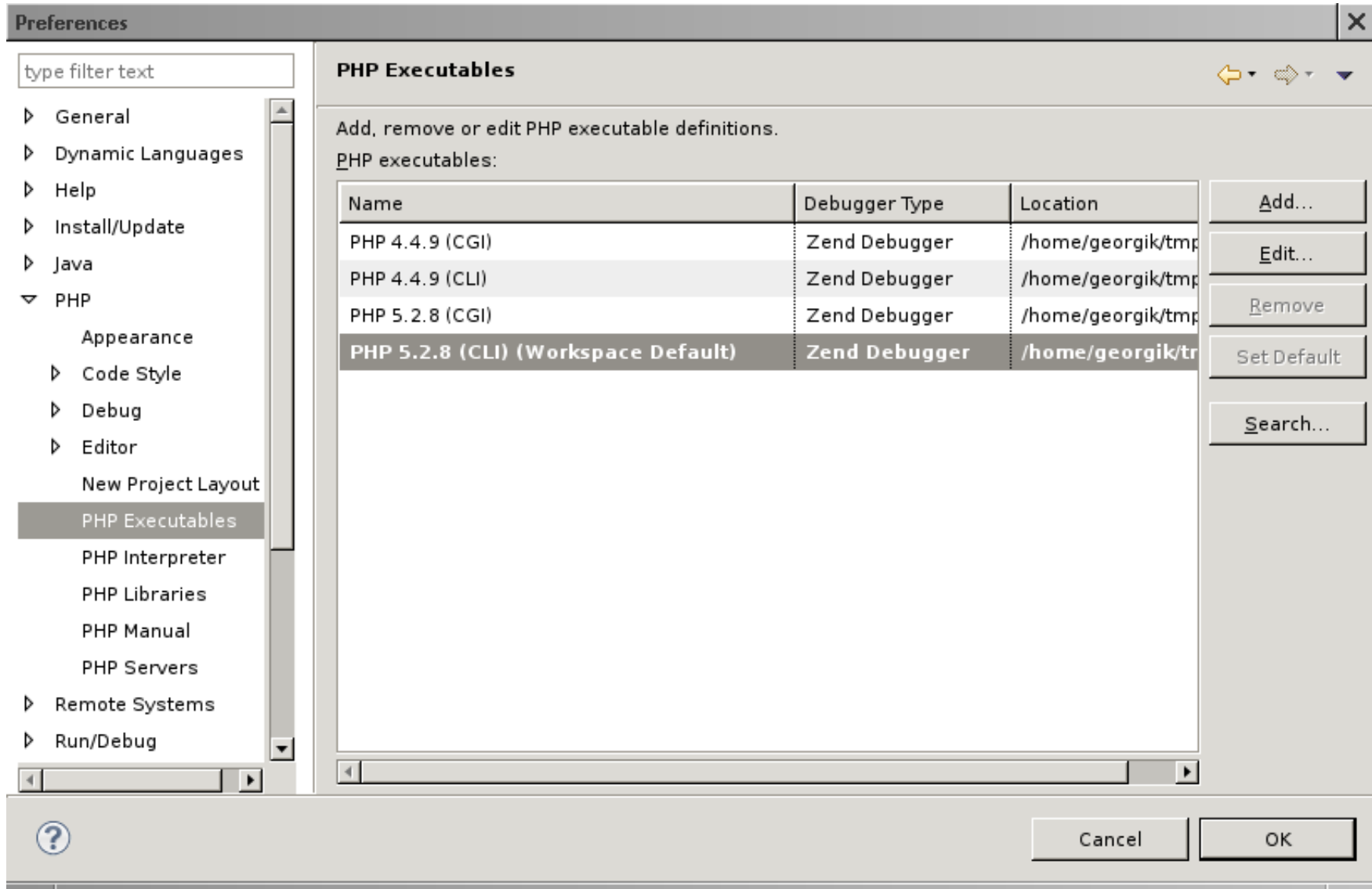
Pozor!
Neobsahuje
Zend Debugger



Dá sa získať z:

<http://www.zend.com/community/pdt>

PHP Zend Debugger



Debug as PHP script

The screenshot shows the Eclipse IDE interface with a PHP script being debugged. The script is paused at line 4, and the Variables window shows the current state of the script's variables.

Debug Window:

- index (2) [PHP Script]
- PHP Application
- index.php (suspended)
- /gombik/index.php at line 4
- /home/georgik/tmp/install/eclipse/eclipse-php-galileo/plugins/org.z

index.php:

```
1 <?php
2
3 $text = "Když něco rozeberem, tak leda ";
4 $engine = array("bludozer", "debugger");
5
6
7 ?>
```

Variables Window:

Name	Value
\$_ENV	Array [59]
\$HTTP_ENV_VARS	Array [59]
\$argv	Array [1]
\$argc	(int) 1
\$_POST	Array [0]
\$HTTP_POST_VARS	Array [0]
\$_GET	Array [0]
\$HTTP_GET_VARS	Array [0]
\$_COOKIE	Array [0]
\$HTTP_COOKIE_VARS	Array [0]
\$_SERVER	Array [64]
\$HTTP_SERVER_VARS	Array [64]
\$_FILES	Array [0]
\$HTTP_POST_FILES	Array [0]
\$_REQUEST	Array [0]
\$text	(string:30) Když něco rozeberem, tak leda

(string:30) Když něco rozeberem, tak leda

Skúsme čo na to Apache

```
index.php http://localhost/~georgik/gombik/index.  
1 <?php  
2  
3 $text = "Když něco rozeberem, tak leda ";  
4 $engine = array("bludozer", "debugger");  
5 $result = $text . $engine . "em!";  
6  
7 print $result;  
8  
9 ?>
```

```
index.php http://localhost/~georgik/gombik/index.  
http://localhost/~georgik/gombik/index.  
Když něco rozeberem, tak leda Arrayem!
```

Nezastavil!

Drobná úprava v php.ini

```
zend_extension=/home/g.../ZendDebugger.so  
zend_debugger.allow_hosts=127.0.0.1  
zend_debugger.expose_remotely=always  
zend_debugger.connector_port = 10013
```

Pohoda s PHP

The image shows a PHP debugger interface with two main panels. The left panel displays the code being executed, and the right panel shows the current state of variables.

Code Editor (Left Panel):

```
1 <?php
2
3 $text = "Když něco rozeberem, tak leda ";
4 $engine = array("bludozer", "debugger");
5 $result = $text . $engine . "em!";
6
7 print $result;
8
9 ?>
```

Variables Panel (Right Panel):

Name	Value
\$_ENV	Array [11]
\$HTTP_ENV_VARS	Array [11]
\$_POST	Array [0]
\$HTTP_POST_VARS	Array [0]
\$_GET	Array [9]
\$HTTP_GET_VARS	Array [9]
\$_COOKIE	Array [0]
\$HTTP_COOKIE_VARS	Array [0]
\$_SERVER	Array [31]
\$HTTP_SERVER_VARS	Array [31]
\$_FILES	Array [0]
\$HTTP_POST_FILES	Array [0]
\$_REQUEST	Array [9]
\$text	(string:30) Když něco rozel
\$engine	Array [2]
\$result	(string:38) Když něco rozel

At the bottom of the Variables panel, the output of the script is visible: (string:38) Když něco rozeberem, tak leda

Debugger - zhrnutie



Debuggovať ide všetko!

Len konfigurácia nástrojov nemusí byť okamžite zrejmá.

Čím trávi vývojár najviac času?

Ladením.

Debuggovaním.

Facebookom ;-)

Zmena paradigmy

Zkuste to bez debuggeru!

Milý Marconi!

- Jára Cimrman,
průkopník v hodu HDD do dálky

Odstránenie debuggovania

Riešenie je jednoduché!

Kód bez chýb!

Unit testy

Kód rozdělíme.

Napíšene automatizované testy.

Mňáá to zdržuje!

„Písanie testov zdržuje od reálneho kódu.“

„Musím rýchlo písať kód, šéf chce po mne funkcie“

„Testy som vypoľ, lebo neustále padali.“



OMG !



Vývojár si neuvedomuje, že

nepísaním testov:

Predražuje svoju prácu.

Oneskoroje dodávku produktu.

Poškodzuje ostatných vo firme.

Znižuje možnosť ďalšieho rozšírenia produktu.

Neposkytuje dokumentáciu k svojmu kódu.

PHPUnit - je to tak jednoduché!

Easy to learn to write.

Easy to write.

Easy to read.

Easy to execute.

Quick to execute.

Isolated.

Composable.

Príklad z PHPUnit

```
<?php
require_once 'PHPUnit/Framework.php';

class StackTest extends PHPUnit_Framework_TestCase
{
    public function testPushAndPop()
    {
        $stack = array();
        $this->assertEquals(0, count($stack));

        array_push($stack, 'foo');
        $this->assertEquals('foo', $stack[count($stack)-1]);
        $this->assertEquals(1, count($stack));

        $this->assertEquals('foo', array_pop($stack));
        $this->assertEquals(0, count($stack));
    }
}
?>
```

Spustenie testu

```
(georgik@traper:pts/10)-----(/tmp)
(1:14:59: )— phpunit test.php -----(Št,okt15)
PHPUnit 3.4.1 by Sebastian Bergmann.

.

Time: 1 second

OK (1 test, 5 assertions)
```

Statická analýza kódu

```
<?php  
$fh = fopen($myFile, 'r');  
print fread($fh);  
fclose($fh)  
?>
```

Security hole

RATS

```
(georgik@traper:pts/10) — (/tmp)
(15:13:z) — rats -l php security-hole.php — (Št, okt15)
Entries in perl database: 33
Entries in python database: 62
Entries in c database: 336
Entries in php database: 55
Analyzing security-hole.php
security-hole.php:2: High: fopen
Argument 1 to this function call should be checked to ensure that it
does not
come from an untrusted source without first verifying that it contain
s nothing
dangerous.

Total lines analyzed: 6
Total time 0.000150 seconds
39999 lines per second
```

Švajčiarský nožík - NetCat

Welcome to the official GNU Netcat project homepage

The GNU Netcat project

test.txt:

GET /index.html HTTP/1.1

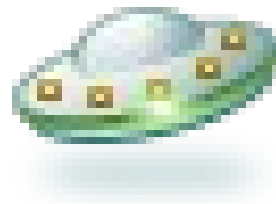
Host: localhost

Connection: close

prázdný riadok

nc localhost 80 <test.txt

Pár zaujímavostí



Software Engineering Radio



<http://www.se-radio.net>

Adrenaline Junkies and Template Zombies

Vzory správania
v projektoch

*Adrenaline Junkies
and Template Zombies*



Understanding Patterns of
Project Behavior

Tom DeMarco, Peter Hruschka
Tim Lister, Steve McMenamin
James Robertson, Suzanne Robertson
Principals of the Atlantic Systems Guild

"Brilliantly insightful." —Howard Look, VP of Software, Pixar



Priestor na otázky.



WebExpo 2009 – Praha

Mgr. Juraj Michálek
SinusGear

Twitter: <http://twitter.com/georgiksk>

Blog: <http://georgik.sinusgear.com>

Ďakujem za pozornosť!



WebExpo 2009 – Praha

Mgr. Juraj Michálek
SinusGear



Twitter: <http://twitter.com/georgiksk>

Blog: <http://georgik.sinusgear.com>