

PHP Quebec 2008 - Montréal, Canada

March 12th, 2008

Derick Rethans and Marcus Börger - dr@ez.no /

marcus@php.net

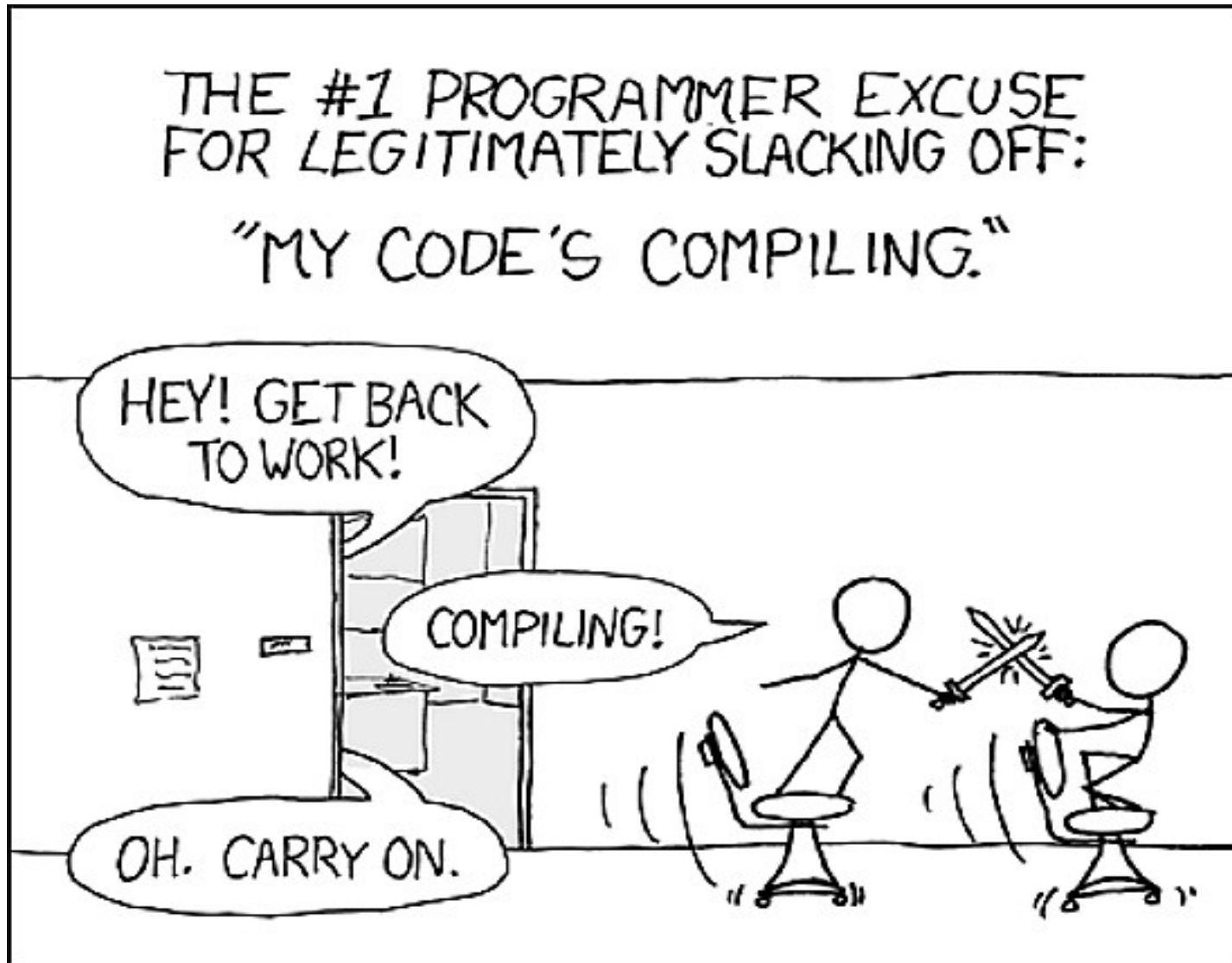
<http://derickrethans.nl/talks.php>

PHP Has No Compiler

Myth #1: PHP Has No Compiler

- Why would you want a compiler?
- Real compiler makes “proper” byte code that can be introspected
- There is no “compiler”, but there is APC
- Only performs caching, and does not store to disk
- APC optimizes, and you don't want that in a normal parser

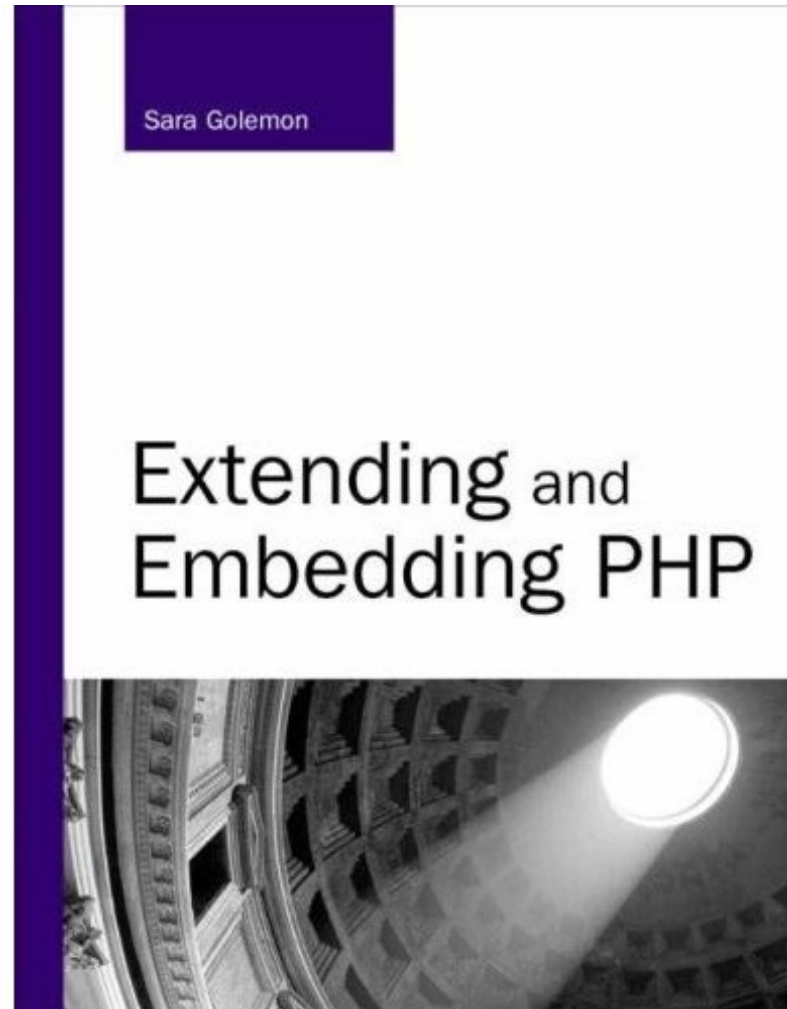
Myth #1: No Compiler



PHP Is Hard To Integrate

Myth #2: PHP Is Hard To Integrate

- As PHP performs badly, another approach is required
- Extensions are hard to write
- You can only write extensions in C
- Who here has tried to write an extension, or considered it?
- There are plenty of extensions, so plenty of examples
- There are also extensions in C++ (SDO, SCA, Colorer...)
- Extensions are hard to use, as you need to recompile PHP
- Shared objects, even closed source exist (Zend, ioncube)
- Documentation is hard to find



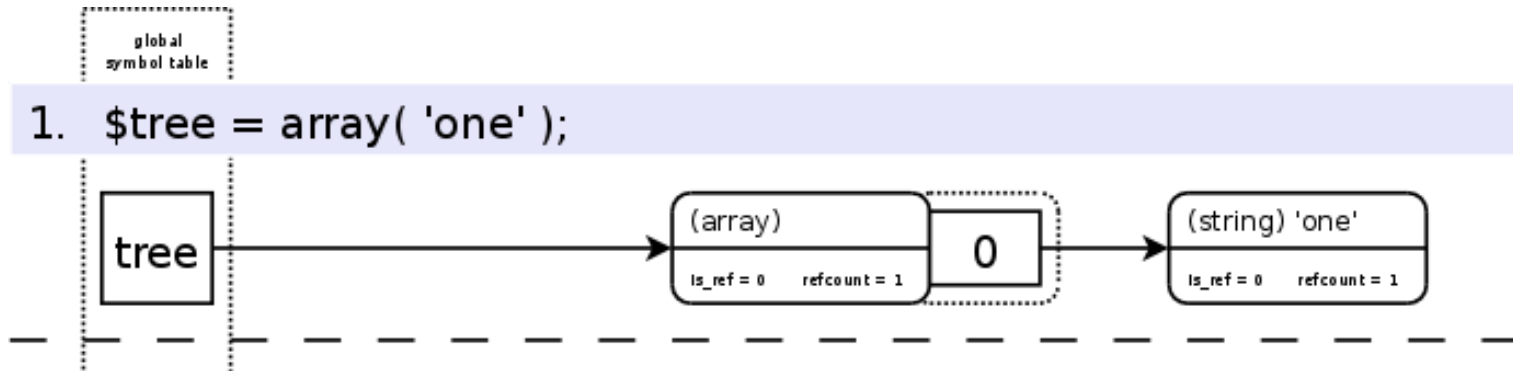
PHP Is An Unstable Platform

- Backwards Compatibility breaks galore
- Who has been affected by this, and which thing?
- We try not to break BC at all, but we're still human
- PHP crashes all the time
- Could just as well been something else... like bad code (infinite recursion)
- And where is the backtrace/bug report?
- But references caused problems!
- Yes, they did – and this was fixed
- It wasn't fixed correctly, as it started throwing notices
- Only if you abused references before
- Why does PHP memory leak?

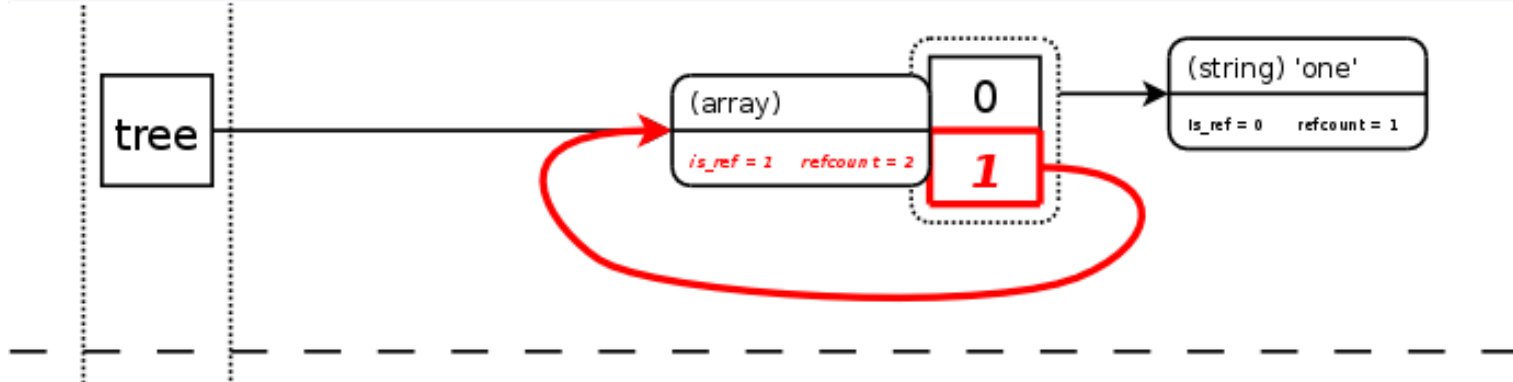
Variables

Circular References

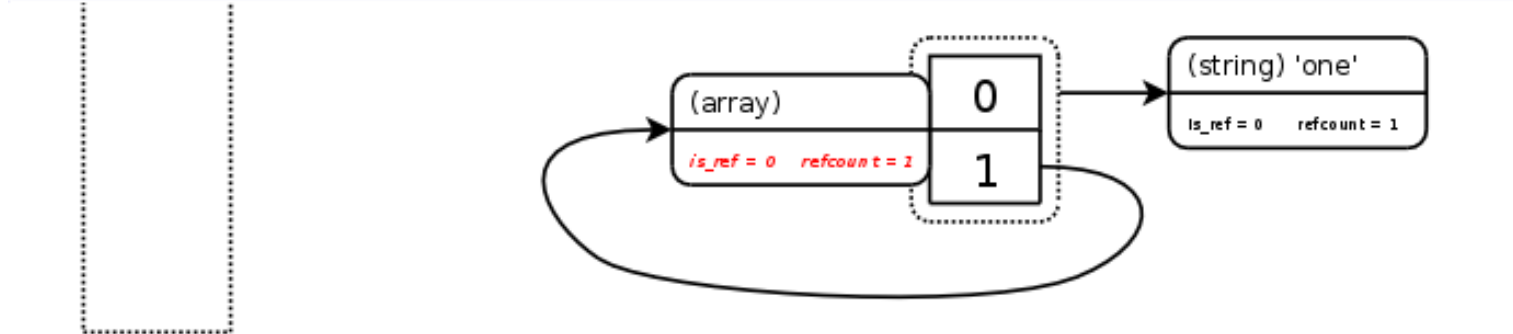
1. `$tree = array('one');`



2. `$tree[] = &$tree;`



3. `unset($tree);`



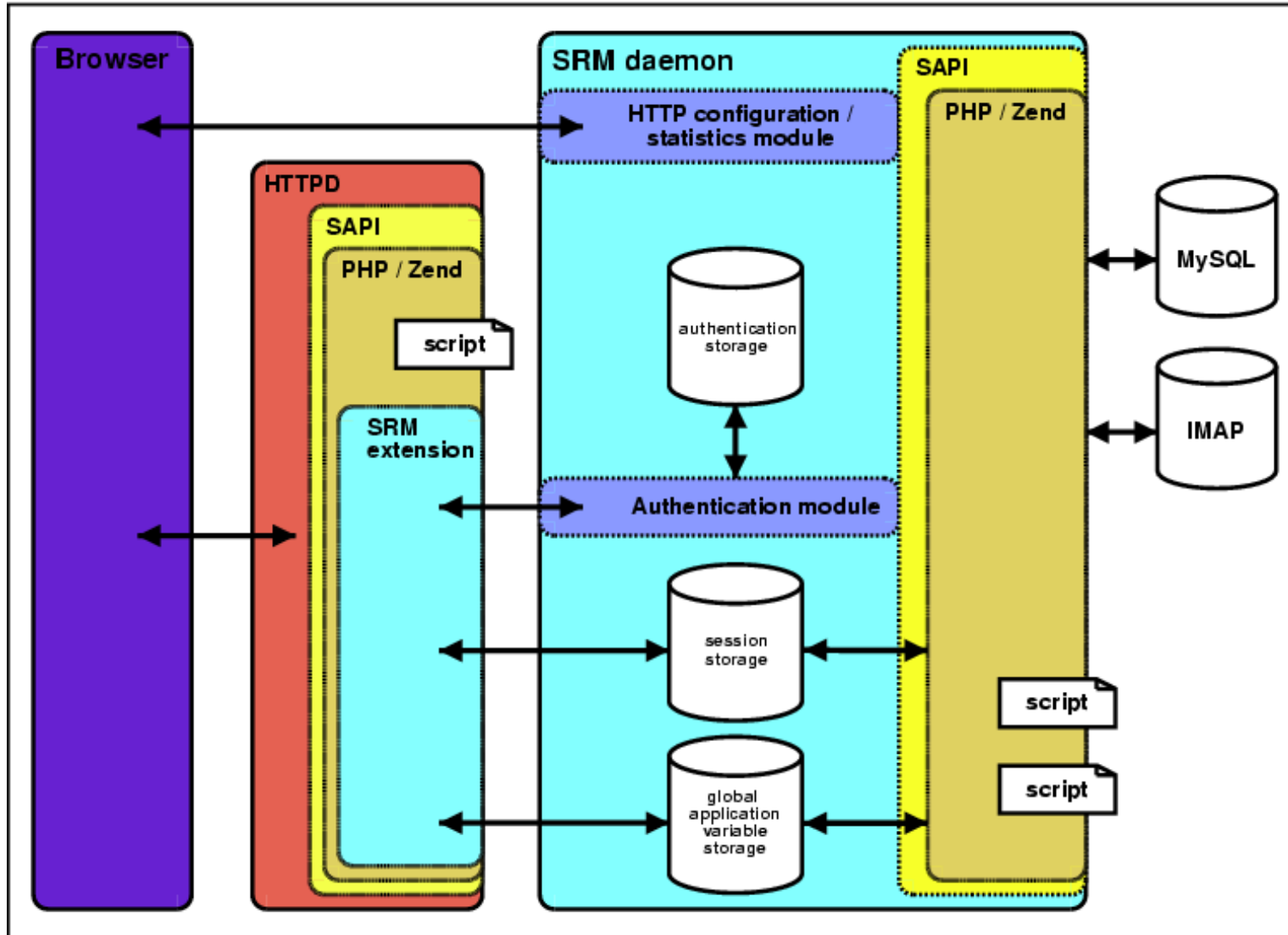
PHP Is Weak(ly typed)

Myth #4: PHP is Weak(ly typed)

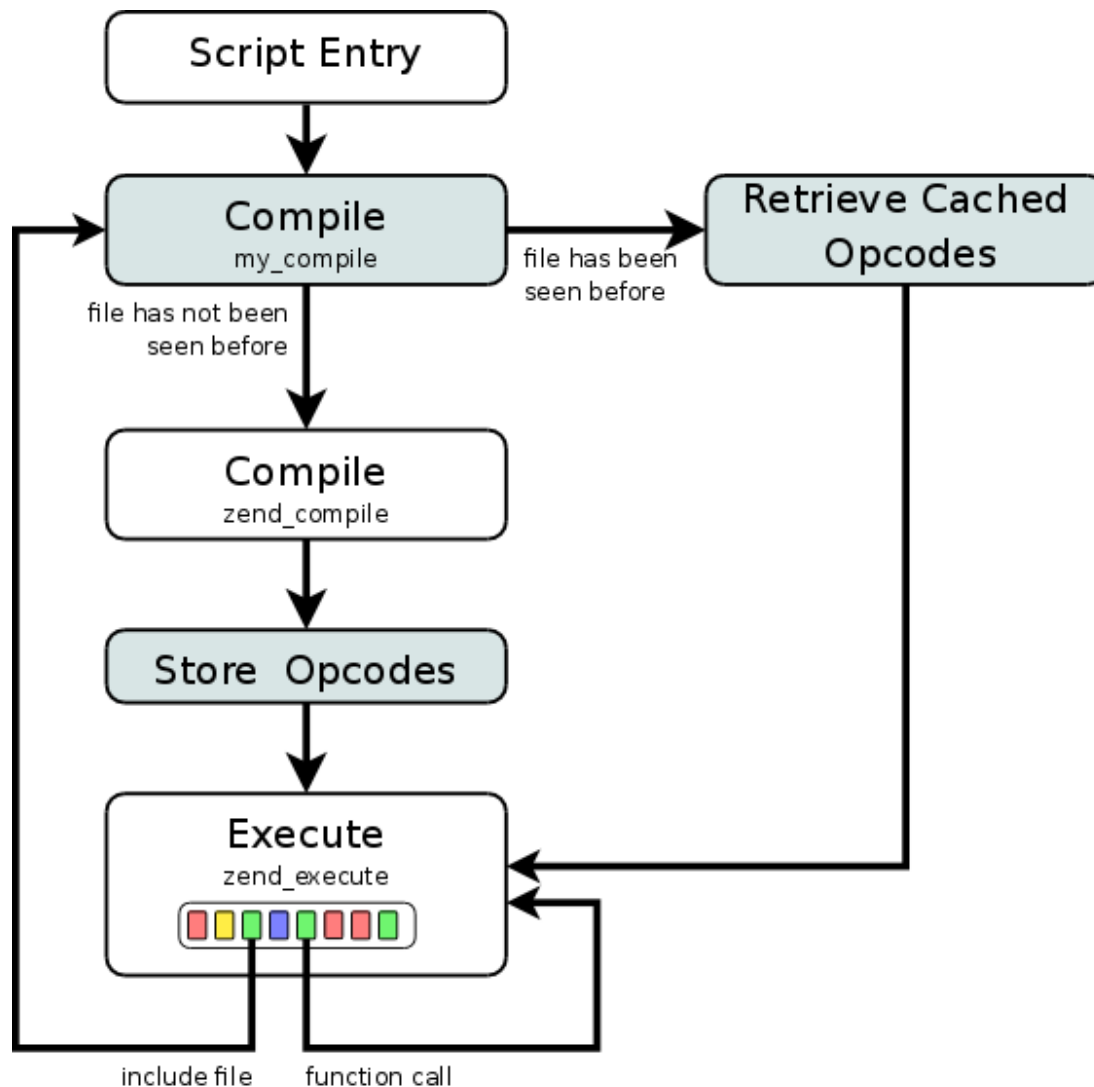
- PHP auto casts variables when it feels like it – because that's how get/post works – strings only.
- Has this every caused you a problem that can not be solved elegantly differently?
- Larger application requires more control and type checks and...
- You can always write type-aware (but not checking) code, saves time and is less code thus faster.
- There is also ext/filter
- Why don't we have scalar type hints?
- Why don't we have auto boxing?

PHP Has No Application Server

Myth #5: PHP Has No Application Server



Myth #5: PHP Has No Application Server



PHP's OO Implementation Is Inferior

Myth #6: PHP's OO Implementation Is Inferior

- You should have implemented the Java model, as that's where everything was stolen from anyway
- PHP also doesn't use OO everywhere where it can.
- PHP is pragmatic, where Java is academic and limiting.
- PHP has to stay backwards compatible
- New extensions also come in OO variants
- OO is not the solution to all your problems
- Why can't we omit \$this?
- Why do we have goto?
- Why is reflection information not cached?

Myth #6: PHP's OO Implementation Is Inferior

- Why don't we have type hints?
- Why don't we have templates?
- Why don't we have generics?
- PHP has no namespaces and packages.
- PHP throws fatal errors and not exceptions for many things
- We have no type hints for properties.
- PHP doesn't have friends
- Why is the visibility checked against the class and not the objects?

- why are there *Recursive and non-Recursive versions of most iterators
- Why is it rewind, and not reset?
- Why do the recursive versions not just do the recursion?

- Why does SPLFileObject not have fread()? Because it's line based.
- Why can't I implement traversable? Very low level shitznitz.
- Why are SplObserver and SplSubject so narrow – dude?

- Why is everything marked as final – because it's safe C-code.
- Why are exceptions not stackable?
- Why don't we have finally?
- Why can't I throw in `__destruct`?
- Why can't I throw an exception from `__autoload()`?

There Are Not Enough Tools For PHP

Myth #7: There Are Not Enough Tools For PHP



Myth #7: There Are Not Enough Tools For PHP

eZ Components

Current view: </home/derick/dev/ezcomponents/trunk/Tree/src>












Date: Fri Sep 7 11:03:39
CEST 2007

Executable lines: 923

Code covered: 99.89%

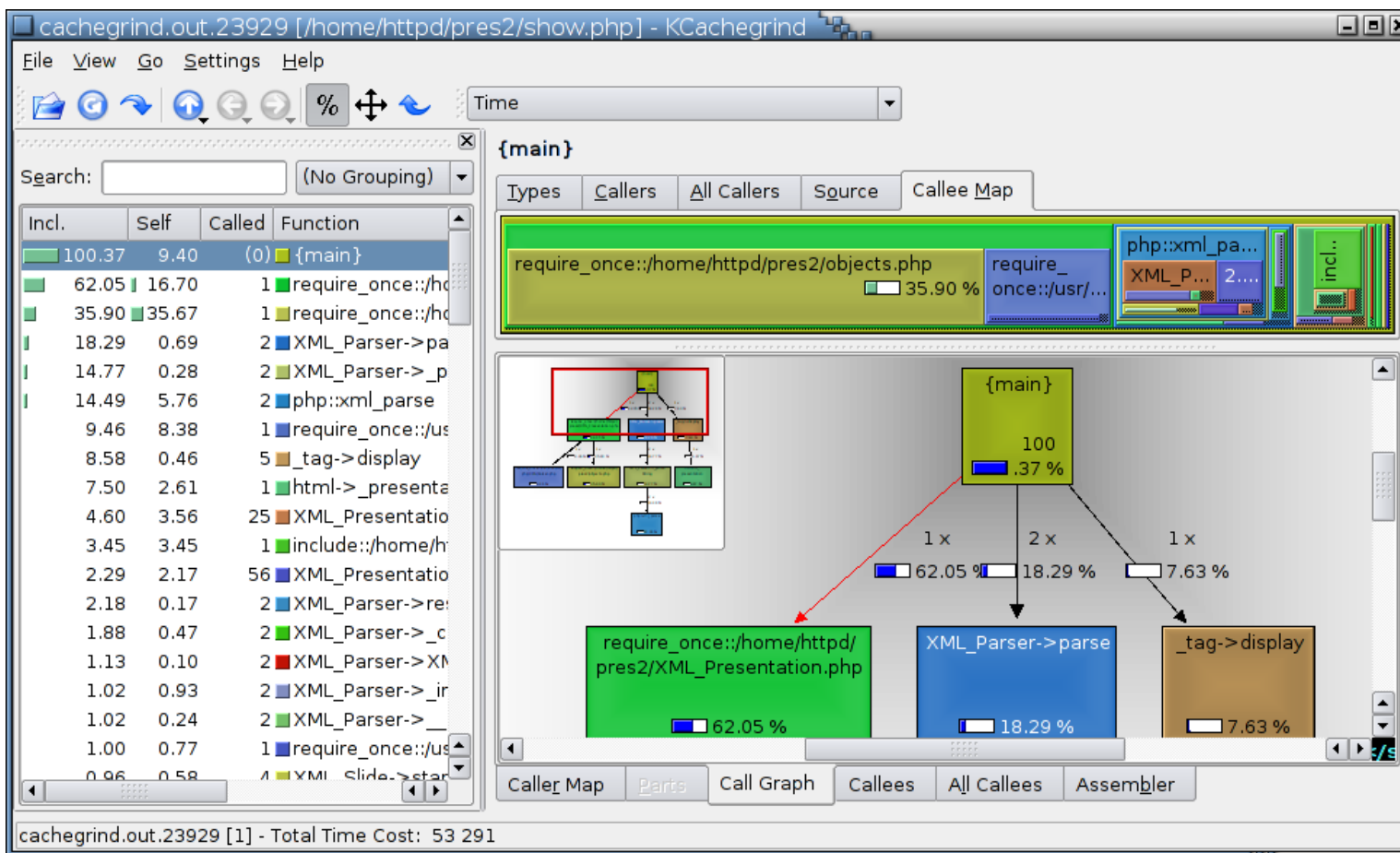
Executed lines: 922

Legend: Low: 0% to 35% Medium: 35% to 70% High: 70% to 100%

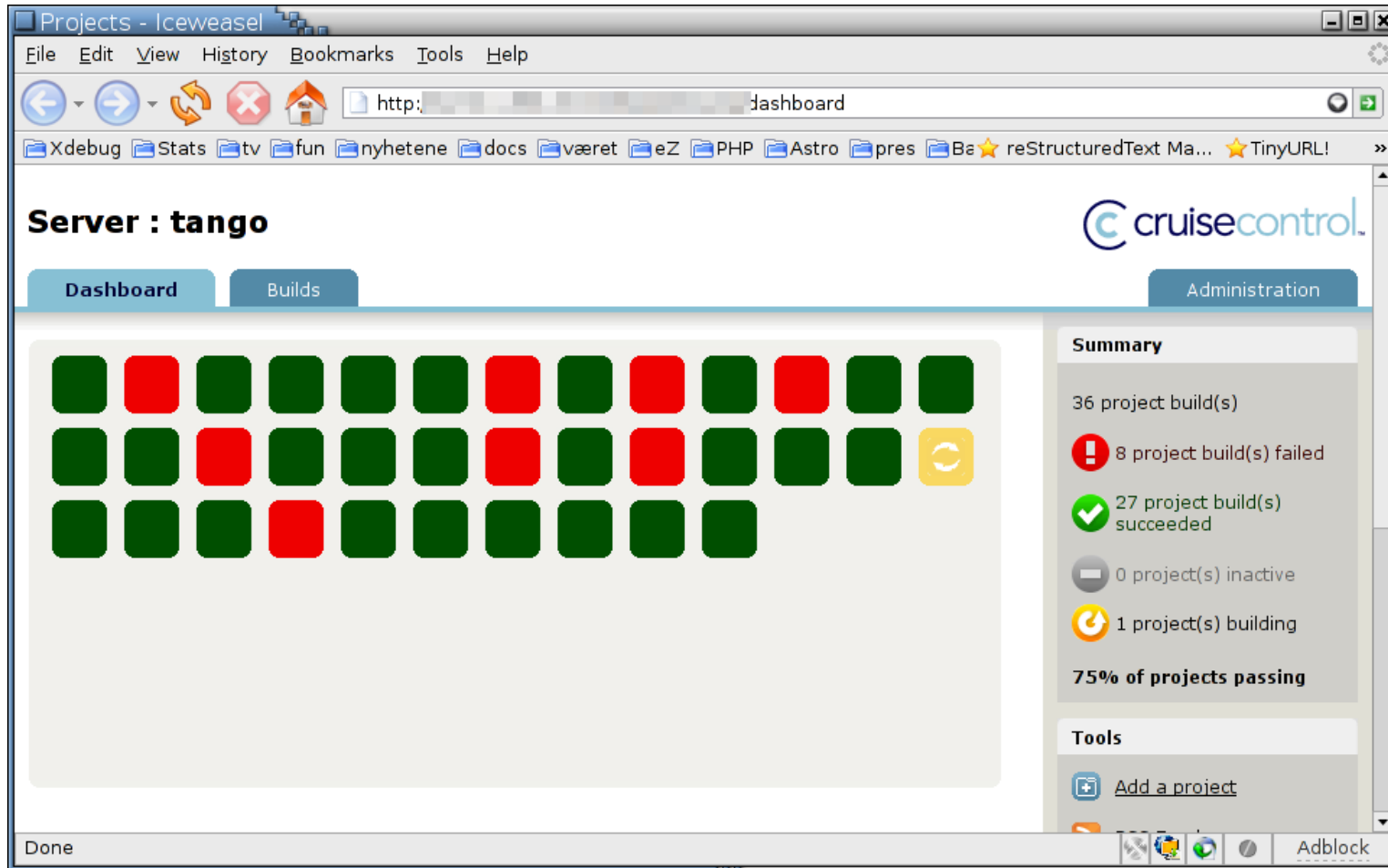
	Coverage (show details)		
backends		100.00%	391 / 391 lines
exceptions		100.00%	42 / 42 lines
interfaces		100.00%	2 / 2 lines
options		100.00%	39 / 39 lines
stores		100.00%	41 / 41 lines
structs		100.00%	13 / 13 lines
visitors		100.00%	180 / 180 lines
tree.php		98.80%	82 / 83 lines
tree_node.php		100.00%	68 / 68 lines
tree_node_list.php		100.00%	38 / 38 lines
tree_node_list_iterator.php		100.00%	26 / 26 lines

Generated by: [PHPUnit 3.1.7](#) and [Xdebug 2.0.1-dev](#).
Logfile: [Plain Text](#) | [TAP](#) | [XML](#) | [Code Coverage](#)

Myth #7: There Are Not Enough Tools For PHP



Myth #7: There Are Not Enough Tools For PHP



The screenshot shows a web browser window titled "Projects - Iceweasel" displaying the CruiseControl dashboard for server "tango". The browser's address bar shows "http://.../dashboard". The dashboard has three tabs: "Dashboard", "Builds", and "Administration". The main content area features a grid of 36 colored squares representing build statuses: 27 green (succeeded), 8 red (failed), and 1 yellow (building). A sidebar on the right provides a "Summary" of the build results:

- 36 project build(s)
- 8 project build(s) failed
- 27 project build(s) succeeded
- 0 project(s) inactive
- 1 project(s) building
- 75% of projects passing

Below the summary is a "Tools" section with a button labeled "Add a project". The browser's status bar at the bottom shows "Done" and an "Adblock" extension icon.

Extending and Embedding PHP: <http://tinyurl.com/24f4j2>
Xdebug: <http://xdebug.org>
eZ Components: <http://components.ez.no>
Questions?: <mailto:dr@ez.no>