

Script Running Machine

Conférence PHP Québec

March 20th, 2003. Montreal, Canada

Derick Rethans <d.rethans@jdimedia.nl>

- o History
- o SRM by example
- o Technology
- o Future

June 8, 1995	PHP Tools 1.0
Oct 17, 1995	PHP/FI 1.92
Mar. 16, 1996	PHP/FI 1.99k
June 16, 1997	PHP/FI 2.0b12
Oct. 29, 1997	PHP 3.0a1
Nov. 12, 1997	PHP/FI 2.0
Dec. 8, 1997	PHP 3.0b1
Jan. 9, 1998	PHP/FI 2.0.1
June 6, 1998	PHP 3.0
July 4, 1998	PHP 3.0.1
Mar. 1, 1999	PHP 3.0.7
July 19, 1999	PHP 4.0b1
Jan. 1, 2000	PHP 3.0.13
Oct. 21, 2000	PHP 3.0.18
May 22, 2000	PHP 4.0
Oct. 11, 2000	PHP 4.0.3
Apr. 30, 2001	PHP 4.0.5
June 23, 2001	PHP 4.0.6
Dec. 10, 2001	PHP 4.1.0
Dec. 26, 2001	PHP 4.1.1
Feb. 27, 2002	PHP 4.1.2
Apr. 22, 2002	PHP 4.2.0
May 13, 2002	PHP 4.2.1
July 22, 2002	PHP 4.2.2
Sep. 6, 2002	PHP 4.2.3
Dec. 27, 2002	PHP 4.3.0
Feb. 17, 2003	PHP 4.3.1
Aug. 17, 2003	PHP 5.0

- o Loops in early versions
- o boring! let's go on with the cool stuff...

Typical script to count the users 'online':

```
<?php
$timeoutseconds = 300;

$timestamp = time();
$timeout = $timestamp - $timeoutseconds;

mysql_connect();
mysql_select_db('users');

$insert = mysql_query(
    "INSERT INTO online VALUES ('$timestamp', '$REMOTE_ADDR', '$PHP_SELF')");

$delete = mysql_query(
    "DELETE FROM online WHERE timestamp < $timeout");

$result = mysql_query(
    "SELECT DISTINCT ip FROM online WHERE file = '$PHP_SELF'");

$users = mysql_num_rows($result);
? >
```

- o 3! queries per page
- o slow
- o inefficient
- o did I say it was slow?

__sleep and __wakeup:

```
// file.class.php
<?php
class File {
    function File($filename) {
        $this->filename = $filename;
        $this->fp = fopen($filename, 'rb');
    }

    function seek($pos) {
        fseek($this->fp, $pos);
    }

    function __sleep() {
        $this->pos = ftell($this->fp);
        return array('fp', 'pos', 'filename');
    }

    function __wakeup() {
        $this->fp = fopen($this->filename, 'rb');
        fseek($this->fp, $this->pos);
    }
}
? >
```

__sleep and __wakeup example:

```
// example1.php
<?php
    require 'file.class.php';
    session_start();

    $f = new File('/etc/hosts');
    $f->seek(20);

    $_SESSION['f'] = $f;
? >
```

```
// example2.php
<?php
    require 'file.class.php';
    session_start();

    var_dump($_SESSION['f']);
? >
```

```

// binsearch.class.php
<?php
class node {
    var $key;
    var $value;
    var $left;
    var $right;
}

class binsearch {
    var $root = NULL;

    function binsearch($elements) {
        foreach ($elements as $key => $value) {
            $this->add_value($key, $value);
        }
        var_export($this->root);
    }

    function add_value($key, $value) {
        $ptr = &$this->root;

        while ($ptr != NULL) {
            if ($key < $ptr->key) {
                $ptr = &$ptr->left;
            } else {
                $ptr = &$ptr->right;
            }
        }
        $ptr = new node;
        $ptr->key = $key;
        $ptr->value = $value;
        $ptr->left = NULL;
        $ptr->right = NULL;
    }

    function get_value($key) {
        $ptr = &$this->root;

        while ($key != $ptr->key && $ptr != NULL) {
            if ($key < $ptr->key) {
                $ptr = &$ptr->left;
            } else {
                $ptr = &$ptr->right;
            }
        }
        return $ptr ? $ptr->value : FALSE;
    }
}
? >

```

```

<?php
    $elements = array (
        'jan'    => 'Dorsten',      'derick' => 'Dieren',
        'stig'   => 'Trondheim',   'dan'    => 'Baltimore',
        'rasmus' => 'San Francisco', 'ilia'   => 'Montreal'
    );

    $tree =& new binsearch($elements);
    echo $tree->get_value('rasmus');
? >

```

```

<?php
class node {
    var $key = 'jan';
    var $value = 'Dorsten';
    var $left =
class node {
    var $key = 'derick';
    var $value = 'Dieren';
    var $left =
class node {
    var $key = 'dan';
    var $value = 'Baltimore';
    var $left = NULL;
    var $right = NULL;
    };
    var $right =
class node {
    var $key = 'ilia';
    var $value = 'Montreal';
    var $left = NULL;
    var $right = NULL;
    };
};
var $right =
class node {
    var $key = 'stig';
    var $value = 'Trondheim';
    var $left =
class node {
    var $key = 'rasmus';
    var $value = 'San Francisco';
    var $left = NULL;
    var $right = NULL;
    };
    var $right = NULL;
};
}
? >

```

- o Functions to work with persistent Objects
- o Bridge between PHP Client and Objects
- o Objects run in threads
- o Manage persistent resources and data



- o Persistent
- o Like J**a beans
- o Compiled only once



- o Oparrays
- o Compiled code
- o One for every script element

- o Opcode
- o Basic execution unit
- o Two operands
- o One result

- o With the SRM daemon
- o UNIX domain sockets
- o `$srm = new SRM ('/var/srm.socket');`
- o TCP/IP sockets
- o `$srm = new SRM ('localhost', 7777);`
- o With Bananas
- o `$binsearch = new SRMApp($srm, 'binsearch');`
- o `$binsearch->function('param1', 2);`

SRM's support for protocols is extensible:

- o Native protocol: fast
- o XML-RPC: easy to use from other languages
- o SOAP: harder to use, but widespread support

```

<?php
class node {
    var $key;
    var $value;
    var $left;
    var $right;
}

class binsearch extends Banana {
    var $root = NULL;

    function binsearch($elements) {
        foreach ($elements as $key => $value) {
            $this->add_value($key, $value);
        }
        var_export($this->root);
    }

    function add_value($key, $value) {
        $ptr = &$amp;this->root;

        while ($ptr != NULL) {
            if ($key < $ptr->key) {
                $ptr = &$amp;ptr->left;
            } else {
                $ptr = &$amp;ptr->right;
            }
        }
        $ptr = new node;
        $ptr->key = $key;
        $ptr->value = $value;
        $ptr->left = NULL;
        $ptr->right = NULL;
        var_dump(func_get_args());
    }

    function get_value($key) {
        $ptr = &$amp;this->root;

        while ($key != $ptr->key && $ptr != NULL) {
            if ($key < $ptr->key) {
                $ptr = &$amp;ptr->left;
            } else {
                $ptr = &$amp;ptr->right;
            }
        }
        var_dump(func_get_args());
        return $ptr ? $ptr->value : FALSE;
    }

    function get_all() {
        var_dump(func_get_args());
        return $this->root;
    }
}

$tree =& new binsearch(array());
$tree->run();
? >

```

Adding a value to the binary tree:

```
<?php
    $s = new SRM('/tmp/srm.socket');
    $t = new SRMApp($s, 'binsearch');

    $t->add_value($argv[1], $argv[2]);
    $all = $t->get_all();

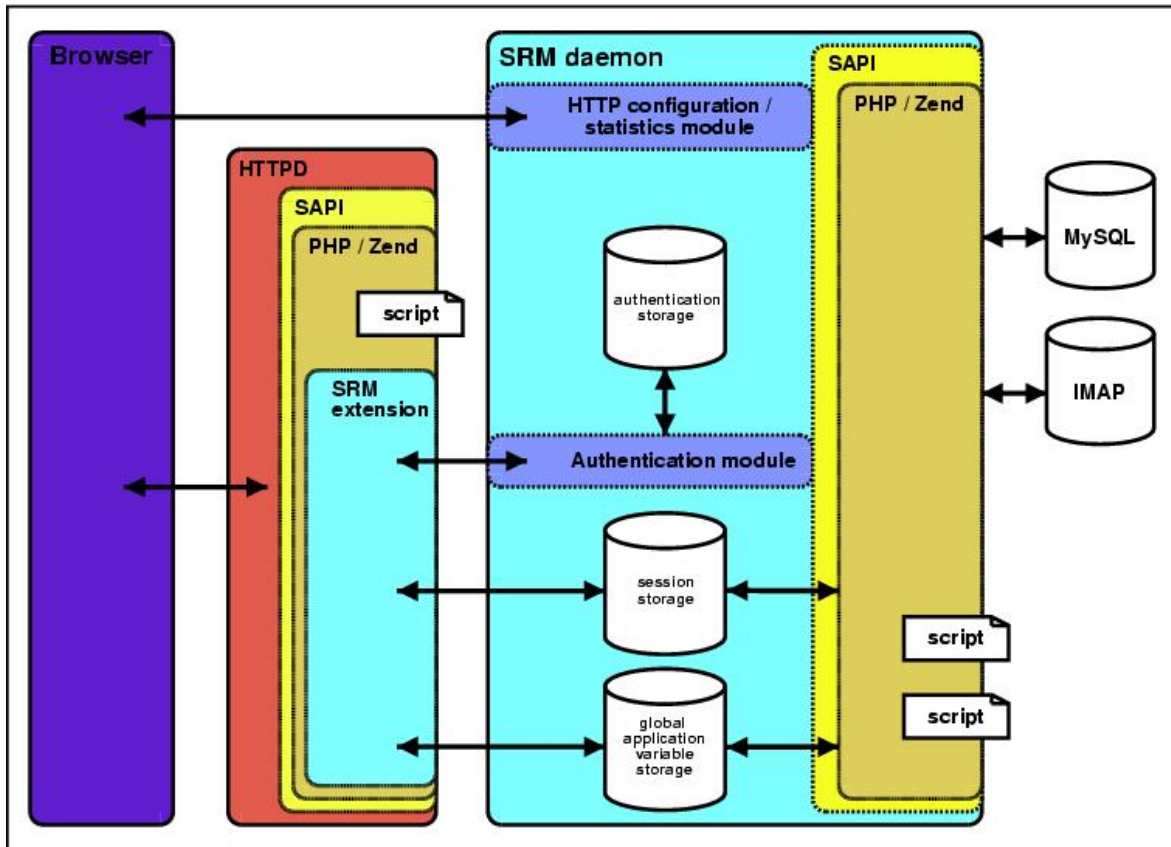
    var_dump($all);
? >
```

Searching for a key in the binary tree:

```
<?php
    $s = new SRM('/tmp/srm.socket');
    $t = new SRMApp($s, 'binsearch');

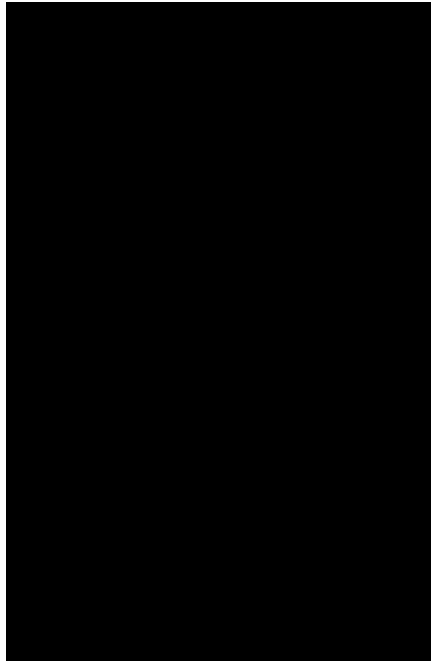
    echo $t->get_value($argv[1])."\n";
? >
```

- o Daemon
- o SAPI for PHP
- o Extension for PHP



- o Periodic calling of functions
- o ACLs to Bananas
- o Load balancing

- o Zend Engine 2
- o `$obj->foo = array (1, 2, 3, 4);`
- o `echo $obj->foo[1];`



FIN



These Slides: <http://pres.derickrethans.nl/srm-montreal>

SRM: <http://www.vl-srm.net>

PHP: <http://www.php.net>

Index

Introduction	2
PHP History (1 of 12)	3
PHP History (2 of 12)	4
The Past: Users on-line	5
The Past: Users on-line	6
Persistent objects	7
Persistent objects	8
The Past: Trees	9
The Past: Trees	10
SRM: Script Running Machine	11
Bananas	12
Oparrays / Opcode	13
Interaction	14
Protocols	15
Persistent Tree	16
Persistent Tree	17
Persistent Tree	18
Parts	19
Design diagram	20
Future	21
Future	22
Questions	23
.....	24
Resources	25