Script Running Machine

Conférence PHP Québec

March 20th, 2003. Montreal, Canada

Derick Rethans <d.rethans@jdimedia.nl>

Introduction

- History
- SRM by example
- Technology
- Future

PHP History (1 of 12)

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

PHP History (2 of 12)

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

The Past: Users on-line

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);
? >
```

The Past: Users on-line

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

Persistent objects

__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);
        }
    }
? >
```

Persistent objects

__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;
        }
    }
? >
```

The Past: Trees

```
<?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;
  };
}
? >
```

SRM: Script Running Machine

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

Bananas

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

Oparrays / Opcode

- Oparrays
- Compiled code
- One for every script element

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

Interaction

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

Protocols

SRM's support for protocols is extensible:

- Native protocol: fast
- XML-RPC: easy to use from other languages
- 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) {
            $ptr = &$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;
            var_dump(func_get_args());
        }

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

            while ($key != $ptr->key && $ptr != NULL) {
                if ($key < $ptr->key) {
                    $ptr = &$ptr->left;
                } else {
                    $ptr = &$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 binairy 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 binairy tree:

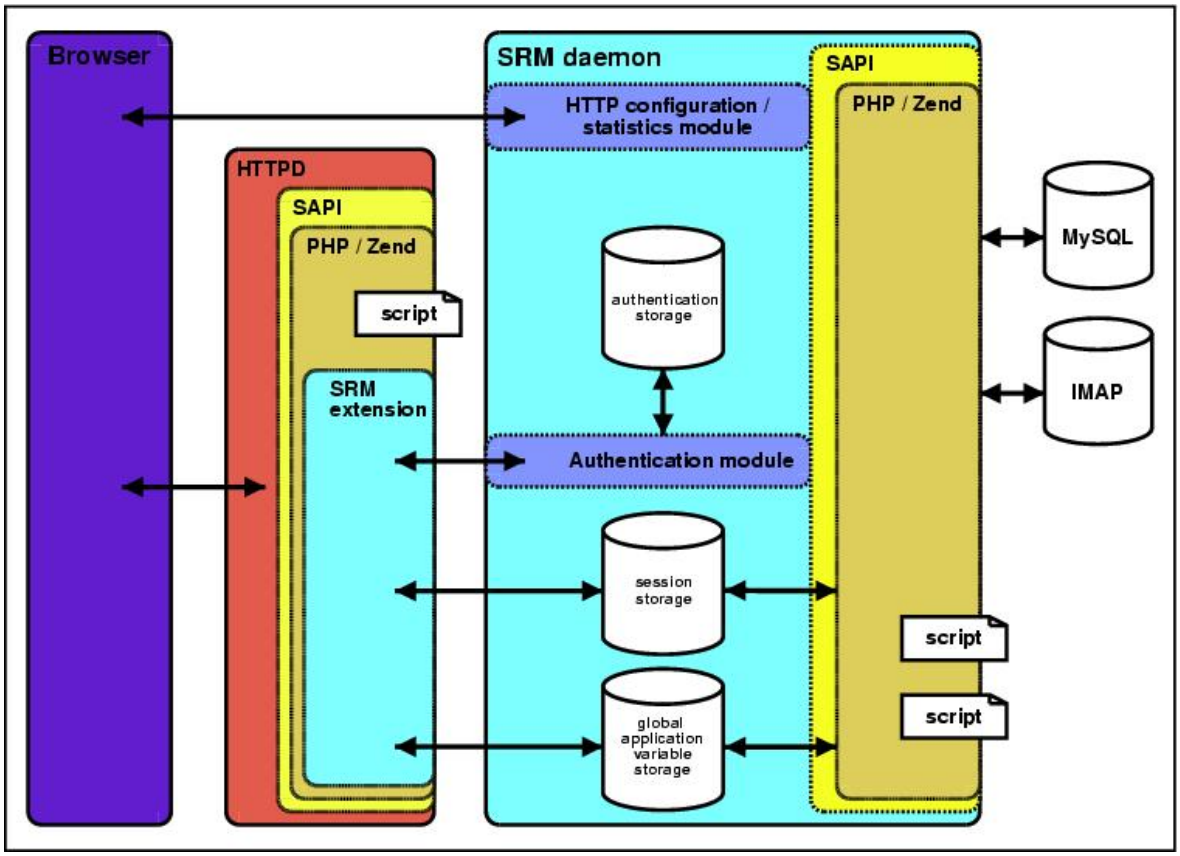
```
<?php
    $s = new SRM('/tmp/srm.socket');
    $t = new SRMApp($s, 'binsearch');

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

Parts

- Daemon
- SAPI for PHP
- Extension for PHP

Design diagram



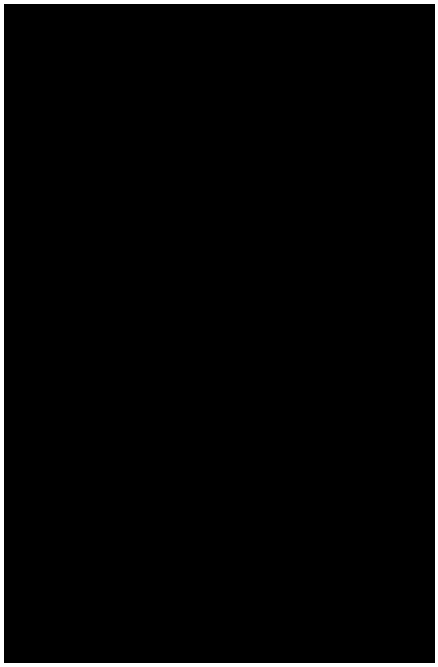
Future

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

Future

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

Questions



FIN

Resources

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

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

PHP: http://www.php.net

Index