On the Origin of Lithium

The framework for people who hate frameworks

CakePHP

CakePHP

CakePHP

I booked this talk with Hans

Lithium

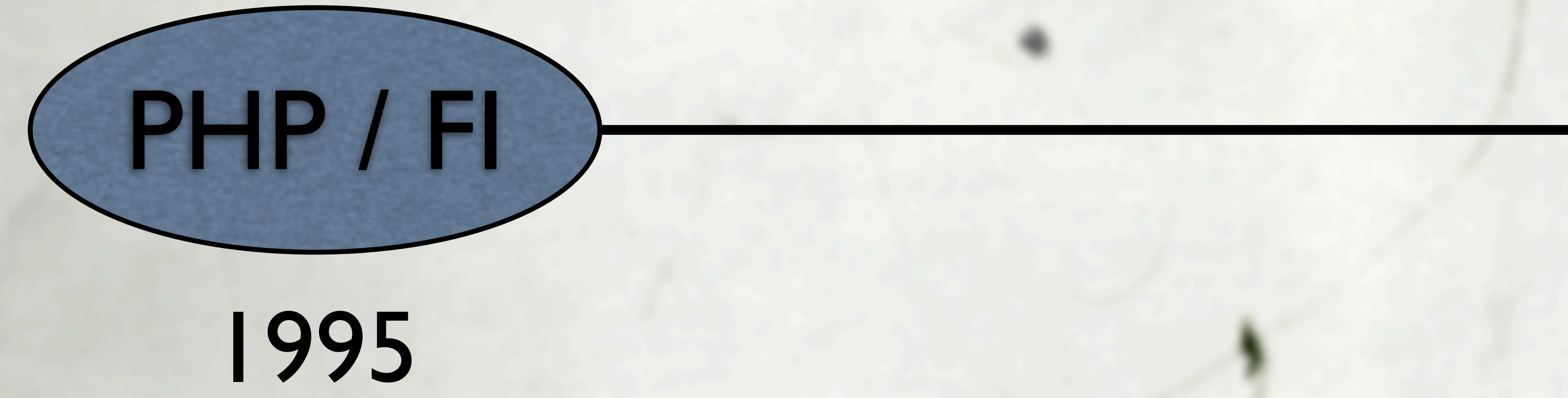
And now here we are

In the beginning...

```
<!--sql database select * from table where user='$username'-->


<!--ifless $numentries 1-->
Sorry, that record does not exist<p>
<!--endif exit-->

Welcome <!--$user-->!<p>
You have <!--$index:0--> credits left in your account.<p>
```



In the beginning...

```
<?
 $name = "bob";
 $db = "db";
 $result = msql($db,"select * from table where firstname='$name'");
 $num = msql_numrows($result);
 echo "$num records found!<p>";
 $i=0;
 while($i<$num);
    echo msql_result($result,$i,"fullname");
    echo "<br>";
    echo msql_result($result,$i,"address");
    echo "<br>";
    $i++;
 endwhile;
>
```

PHP 3

1998

In the beginning...

```
require_once 'MDB2.php';
require_once 'DB/Table.php';
require_once 'Guestboook_Table.php';

$dsn = "phptype://username:password@localhost/database";
$conn = MDB2::connect($dsn);

$table  = 'GuestBook';
$create = 'safe';
$GuestBook =& new GuestBook_Table($conn, $table, $create);

if ($GuestBook->error) {
     echo "Failure!  Try again.";
     print_r($GuestBook->error);
}
```

PEAR

2000

In the beginning...

```
class A {
    function foo() {
        if (isset($this)) {
            echo '$this is defined (' . get_class($this). ")\n";
        } else {
            echo "\$this is not defined.\n";
        }
    }
}
class B {
    function bar() {
        A::foo();
    }
}
```

PHP 4

2000

In the beginning...

```
class A {

    protected $_foo = 0;

    public static function foo() {
        self::$_foo++;
    }
}
```

PHP 5

2004

In the beginning...

```
class PostsController extends AppController {

    public function index() {
        $posts = $this->Post->find(“all”);
        $this->set(compact(‘posts’));
    }
}
```

CakePHP

2005

In the beginning...

```php
class mymoduleActions extends sfActions
{
  public function executeIndex()
  {
    // Retrieving request parameters
    $password    = $this->getRequestParameter('password');

    // Retrieving controller information
    $moduleName  = $this->getModuleName();
    $actionName  = $this->getActionName();

    // Retrieving framework core objects
    $request     = $this->getRequest();
    $userSession = $this->getUser();
    $response    = $this->getResponse();
```

Symfony

2005

In the beginning...

```
class Blogmodel extends Model {

    var $title   = '';
    var $content = '';
    var $date    = '';

    function Blogmodel()
    {
        // Call the Model constructor
        parent::Model();
    }

    function get_last_ten_entries()
```

CodeIgniter

2006

In the beginning...

```
< ?php

class Bootstrap extends Zend_Application_Bootstrap_Bootstrap
{
    protected function _initDoctype()
        {
            $this->bootstrap('view');
            $view = $this->getResource('view');
            $view->doctype('XHTML1_STRICT');
        }

    protected function _initAutoload()
        {
            $autoloader = new Zend_Application_Module_Autoloader
(array(
```

Zend

2007

What have we learned?

- Uniformity: +
- Tight coupling: -
- Lack of extensibility: -

What have we learned?

- People dislike complexity for its own sake
- Things are reactionary (high level)
- A lot of things are superficial (high level)

Ch-Ch-Ch-Changes

- Late Static Binding
- Namespaces
- Closures

Late Static Binding

- Proper subclassing of static classes... finally
- Warm, fuzzy feelings of architectural purity
- Handling state (vs. statelessness)

Namespaces

- Formal, non-hacky way to organize classes
- No_More_Class_Names_That_Go On_For_Days
- New PEAR-inspired naming standard

Standards Adopters

- Agavi
- Symfony
- Doctrine
- PEAR
- Solar
- Zend Framework
- ...

```
vendor\package\Foo = "vendor/package/Foo.php"
```

Closures

```
function Y($F) {
   return current(array(function($f) { return $f($f); }))->__invoke(function($f) use ($F) {
        return $F(function($x) use ($f) {
            return $f($f)->__invoke($x);
        });
    });
}
```